MARCH 1976 $1.50

BYTE

the small systems journal



Cassette Interfaces
Microprocessor Update: CP1600
More on Algebraic Expressions
Hand Assembly
Space Ace

In the Queue

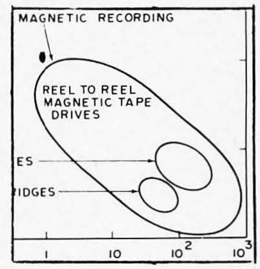
BYTE #7

MARCH 1976

Foreground

Background

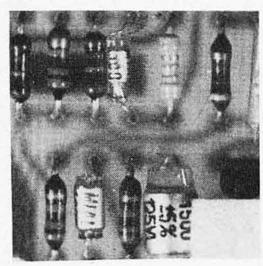
Nucleus



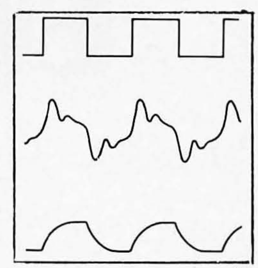
p. 18



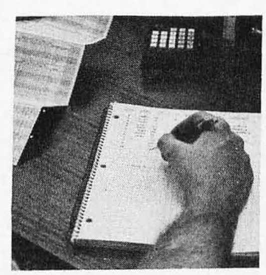
p. 29



p. 30



p. 40



p. 52

BYTE magazine is published monthly by BYTE Publications, Inc., 70 Main St, Peterborough, New Hampshire 03458. Subscription rates are $12 for one year worldwide. Two years, $22. Three years, $30. Second class postage application pending at Peterborough New Hampshire 03458 and at additional mailing offices. Phone 603-924-7217. Entire contents copyright 1976 BYTE Publications, Inc, Peterborough NH 03458. Address editorial correspondence to Editor, BYTE, 70 Main St, Peterborough NH 03458.

New BYTE phone: 603-924-7217

In This BYTE

One result of the BYTE Audio Cassette Symposium last November was a provisional standard for audio recording, essentially identical to that described by Don Lancaster in issue No. 1 of BYTE last September. In this issue, Don presents an updated design and describes how to **Build The Bit Boffer.** Then, to show that there's more than one way to spin a tape, Harold Mauch gives some details of a second system compatible with the standard in his article **Digital Data on Cassette Recorders.**

Jack Hemenway uses Don Lancaster's Bit Boffer design to wire up **The COMPLEAT Tape Cassette Interface** using an ACIA attached to his 6800. But completeness requires more than hardware, so Jack provides software of six subroutines for the control and transfer of data with this interface.

Creating programs can be done with a variety of tools. The best way is to use an interactive terminal with mass storage available and a good high level language. But, when you move up country or have other reasons to be away from convenient access to monster machines, at first things have to be done the hard way using a new home brew machine. This makes the techniques of **Assembling Programs by Hand** invaluable for your bag of tricks.

In the previous issue, information on joysticks and slide pots was presented. In this issue, John Schulein supplies a short note on another **Pot Position Digitizing Idea.**

It is one thing to tell how to do something, but why it works is often a separate topic. William A Manly provides answers to a lot of the "whys" of magnetic mass storage in his article on the physics of **Magnetic Recording for Computers.**

One of the newer microprocessor designs is the General Instrument CP1600. In his **Microprocessor Update,** Bob Baker summarizes the technical information about this chip design.

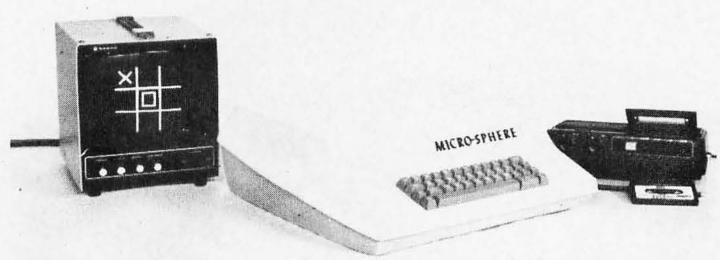
Last month, Prof W Douglas Maurer described some salient points about processing algebraic expressions. In this issue, he continues the discussion with Part 2 of **Processing Algebraic Expressions.** This includes a simplified explanation of what it means to generate code as in a compiler.

Peek inside a video display terminal with Don Walters' quick summary of **What's in a Video Display Terminal.**

And to round out the theme of magnetic recording technology, the cover shows a typical Philips style audio cartridge.

Announcing!! The world's most advanced, low cost, computer system available today!!

The MICRO-SPHERE 200



SPECIAL FEATURES: $860

- **Completely Assembled and Tested (Ready to use; not a kit.)**
- **4k RAM (EXPANDABLE TO 8k)**
- **CASSETTE LOADER**
- **Sphere Cassette Operating System (SCOS) supports file handling and Advanced Program Development, System, an aid for developing your programs.**
- **16 Line by 21 Character Alpha Numeric Character Generator (128 by 128 B & W Dot Matrix Graphic System**
- **Monte Carlo Games Package**
- **Operators Manual**
- **Attractive Mar-Resistant Plastic Case**

The Micro-Sphere 200 Series computer is the most ADVANCED, low-cost computer SYSTEM available today. Together with a TV and up to three cassette recorders you can have big computer performance at a rock bottom price.

The system features a 6800 type micro-computer with 4000 characters (4K Bytes) of internal Random Access Memory (RAM) Storage. The memory is easily expandable to 8000 total characters with the addition of an optional second 4K of RAM, with even more memory to be made available shortly. The 4K bytes of memory is equivalent to 6-8 pages of close typewritten material.

Access into the Micro-Sphere is achieved by the keyboard or from cassette recorders. The computer can display information on a standard TV screen (optionally supplied) or store information on a cassette recorder.

The keyboard uses highly reliable keyswitches to insure user satisfaction. It is full alpha-numeric including an integrated numeric key pad. The cassette interface uses the "Kansas City" standard which means that you can use even the least expensive cassette recorders with your system satisfactorily though we suggest that you do use high quality tapes with your system.

You may use your own TV without modification as the system display device, or you may purchase one from Sphere.

The power requirement is a single 110 volt AC outlet. The unit uses less power than an ordinary 100 watt light bulb. All fuses, jacks, switches, and interface signals are provided on an easy-access panel at the rear of the cabinet.

The Micro-Sphere is supplied with a built-in loading program from cassette, which is in one of the several standard or optional Read Only Memory (ROM) Integrated Circuits (IC).

ROM IC's are pre-programmed with specific non-eraseable information. This feature greatly reduces program loading time and inconvenience, ROM's also save valuable RAM storage. RAM's lose all stored memory whenever the computer power is turned off, while ROMs retain all programs indefinitely.

The Micro-Sphere is unique in that 16,384 different dots on your TV screen can form any number of pictures or designs which you have instructed your computer to display. These images can be changed by the computer program at a rate that appears as real-life movement, such as aircraft flight simulation, "walk-through" inspection of architectual mock-ups, time-lapse stock market graphic analysis, or even computer generated art forms, or space flight simulation where you can guide your spacecraft to the moon, planets or the universe.

You can experience the thrill of rolls, dives, loops, near-collisions and other types of aerial maneuvers to out wit the Red Baron in your Sopwith Camel and then the excitement of a victorious landing as you crash at the end of the approaching runway.
the end of the approaching runway.

This same 128 row by 128 column dot matrix can form an alpha-numeric display of up to 16 lines by 21 characters. An optional graphics input device (Mouse) digitizes hand movements when moved about on a flat surface. The "Mouse" has a window and crosshairs, so it may be used for the accurate entry of maps or other graphic data.

The mouse may also be used in the place of a joy stick for flight simulation or to enter hand movements for ping-pong or other games of skill.

SPHERE CORPORATION

791 South 500 West Dept. 130 (801) 292-8466
Bountiful, Utah 84010

No.	Description		Price/Each	Total
200	MICRO-SPHERE 200 - SYSTEM PRICE INCLUDES "A" ITEMS BELOW		$ 860.00	$ 860.00
	"A" ITEMS: 6800 type Micro-Processor unit			
	4K of Memory (RAM)			
	Cassette Loading System (ROM)			
	Sphere Cassette Operating System (SCOS) Cassette — Includes Floating Point and Trig Package	1 time license fee@ $137.50; Cassette copy @ 12.50	$150.00	Incl.
	Monte Carlo Games Package (Cassette)		$10.00	Incl.
	First Cassette Interface			
	128 by 128 B&W Dot Matrix Graphics Display			
	Alpha-Numeric Keyboard			
	Attractive Mar-Resistant Plastic Case			
	Operators Manual			
	OPTIONS AVAILABLE THROUGH FACTORY INSTALLATION. * To install options after purchase is $35.00 per shipment to our plant.			
	"B" ITEMS: Second 4K of memory (RAM)		$180.00	$180.00
	Character Generator (ROM)		$25.00	$25.00
	"C" ITEMS: Second Cassette Interface		$50.00	$50.00
	Extended Business Basic (ROM) Includes Business Basic Manual		$400.00	$400.00
	Floating point & Trig package (ROM)		$130.00	$130.00
	Third Cassette Interface		$50.00	$50.00
	OPTIONS FOR PURCHASE NOT NEEDING FACTORY INSTALLATION.			
	Extended Business Basic on Cassette (Requires 2nd 4K of RAM and Character Generator in ROM.) Includes Business Basic Manual, Floating Point & Trig Package		$100.00	$100.00
	9" TV for use with Micro-Sphere 200		$150.00	$150.00
	"Mouse" Graphics Input Device (Available in May 1976)	2 ea.	$150.00	$150.00
	Operators Manual (SCOS)		$10.00	$10.00
	Business Basic Manual		$10.00	$10.00
	Maintenance Manual		$40.00	$40.00
	Empty Cassette Tapes	3 for	$10.00	$10.00
200A	INCLUDES MICRO-SPHERE 200 PLUS ALL OF "A" ITEMS ABOVE REGULAR $860.00 -		$860.00	$860.00
200B	INCLUDES MICRO-SPHERE 200 PLUS ALL OF "A" & "B" ITEMS ABOVE REGULAR $1215.00		$1215.00	$1215.00
200C	INCLUDES MICRO-SPHERE 200 PLUS ALL OF "A" & "B" & "C" ITEMS ABOVE REGULAR $1645.00 - EVERYTHING IS IN ROM !!		$1645.00	$1645.00

ALL UNITS ARE COMPLETELY ASSEMBLED AND READY TO USE!!

SPECIAL INTRODUCTORY ORDER FORM

Item	Description	Quantity	Price/Each	Total
1				
2				
3				
4				

NAME ______________________
please print clearly

STREET ______________________

STATE ______________________

CITY: ______________________
STATE ZIP

PHONE NO. ______________________

BANK CARD NO. ______________________

SIGNATURE ______________________

A. Item Purchase Total	
B. Utah Residents add 4.75% tax	
C. Postage, handling, shipping and insurance add 2% of A	
D. Full Warranty = 10% of A.	
E. Order Total	
F. Down Payment = 25% of E.	
G. COD Balance	

SPHERE generally offers 60-90 day delivery on its products, however, parts availability may delay delivery beyond that time.

Orders may be cancelled after 120 days without penalty. Spheres only obligation is to deliver the product. Introductory offer valid in U.S.A. only.

SPHERE CORP. 791 South 500 West, Bountiful, Utah 84010 Tel. (801) 292-8466

Magnetic Recording Technology

Editorial by Carl Helmers

No computer system is complete without mass storage of some form. Mass storage is a nice subject to talk about, but just what is it and how can it be implemented? What are the factors which affect the ability of your computer to dump and recover data in machine readable form? What are the technologies available at prices low enough to be affordable by many individuals? How is a tape interface controlled?

Much of this issue of BYTE is devoted to articles on the problem of mass storage and practical but inexpensive systems. In particular the theme is using Philips style audio recording cassettes as an inexpensive but readily available magnetic recording technology. While no claim can be made for complete coverage of every topic in the area of magnetic recording for small scale computing systems, readers should find a high concentration of personally useful information in this issue: two articles on specific hardware means of implementing the BYTE Audio Cassette Symposium's provisional standard, one article on software interfaces for control of a typical interface, and an article on the physics of magnetic recording technology. Then there are of course several short subjects on the same theme.

Mass Storage – What Is It?

When you purchase a machine with (for example) 4096 bytes of memory, you have the ability to store a total data content of 4096 bytes, no more. The particular number of bytes in your system is arbitrary – the fact that your random access memory is limited is always present. If this were all the memory you had, the systems and programming you could accomplish would sooner or later be limited. At any given point in time you might want to run programs requiring less than the 4096 byte limit; but it is quite likely that you'll want to have more than one such program as you build up your software repertoire.

Similarly, if all you have is main memory, what happens when the power goes off by intention or accident? Data in the semiconductor memories used in most kits typically will not survive the loss of power.

How do you save programs against the possibility of power failure or in order to allow use of different software from time to time? The technology of off line mass storage was created to answer this question. Use of off line storage is a requirement of virtually any programmable computer system. You see it in the smallest HP-65 and SR-52 hand calculators, and in the largest of large scale systems presently in existence.

One definition of mass storage is simply, a method of storing larger amounts of data (in total) than could fit into the main memory of a given processor at one time. On large scale systems, this is usually done using disk drives and drum memories as well as other high speed high capacity random access IO devices. The typical medium priced minicomputer system uses a hard surface magnetic disk drive as its mass storage device; a removable cartridge for at least one spindle allows off line copies of data to be retained. The low priced minicomputers and high priced microcomputer systems typically use floppy disk technology at the present time. And of course the programmable calculators use miniature magnetic card IO devices to store many more programs than could possibly fit in the calculator at one time.

So what does this leave open to the low priced computer amateur's general purpose system? The answer of course is the use of audio tape technology as described by several of the articles in this issue. It is used to accomplish the mass storage function of the larger systems in an engineering tradeoff of access time against total cost. The result is a low speed but inexpensive off line storage method.

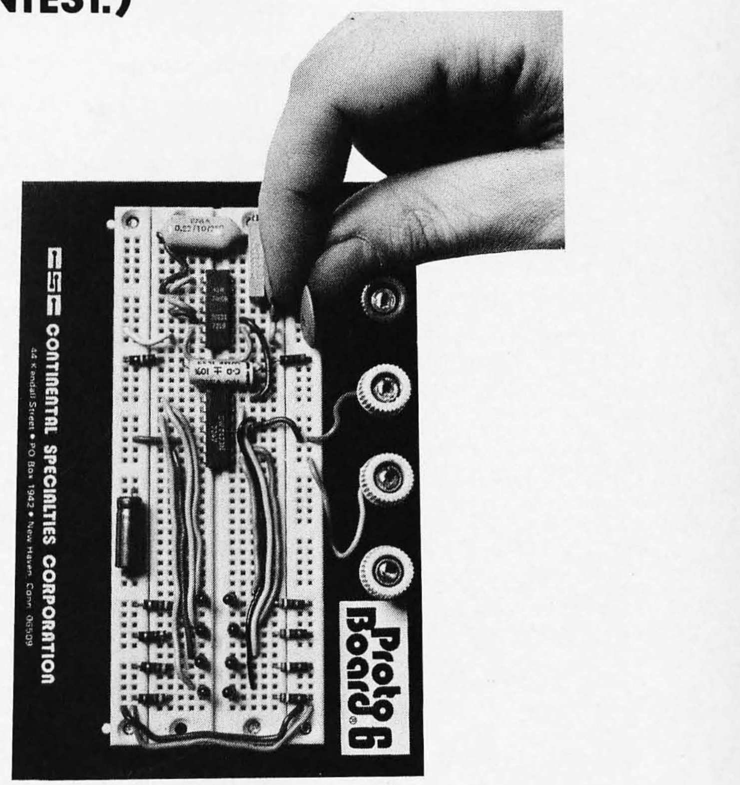
What Do You Use Mass Storage For?

The basic reason for off line mass storage was outlined in the previous section: making up for the limitations of a finite main

THE SOLDERING IRON VS. THE PROTO-BOARD®

(IT'S NO CONTEST.)





The Soldering Iron

If you're still designing and testing breadboard circuits the conventional way, you're doing a lot of extra work, and getting a lot of grief in return. You've got to think as much about manual labor as you do about the circuit. Maybe even more.

Every time you add a component, there are 2 or more connections to make... over a dozen with most IC's, while watching out for overheating components and cold solder joints. And that's only half the problem.

The other half is when you want to change components or connections. Even with good desoldering equipment, you can still have a hassle on your hands. (Ever try to desolder a temperature-sensitive 14-pin DIP on a component-filled board?)

Solder. Desolder. Resolder. Desolder: Now there's a better way...

The Proto-Board

With the CSC Proto-Board breadboarding system, connecting components is as simple as pushing a lead into a hole. Rugged 5-point contacts insure low-resistance connections, and where jumpers are needed, components are interconnected with standard #22 AWG solid wire.

That's all there is to it.

You can choose Proto-Boards with anywhere from 630 to 3060 solderless tie-points. Proto-Boards with or without regulated power supplies. Even assemble your own, with the same solderless QT sockets and Bus Strips,* for smaller (or larger) capacity. However you do it, you'll save time... money... aggravation ...on every circuit.

For more information, see your CSC dealer or distributor... or contact us for our catalog and distributor list.

CONTINENTAL SPECIALTIES CORPORATION

CSC

Continental Specialties Corp., 44 Kendall Street, Box 1942, New Haven, CT. 06509 • 203-624-3103 TWX: 710-465-1227
West Coast Office: Box 7809, San Francisco, CA. 94119 • 415-421-8872 TWX: 910-372-7992
Canada: Len Finkler Ltd., 25 Toro Rd., Downsview, Ontario • 416-630-9103

*U.S. Pat. No. D235,554

memory in any particular computer's implementation. The uses of the mass storage are basically similar to the uses of memory within the computer: mass storage contains program and data information. The problems of using and coordinating the use of the mass storage facility are independent of the information content of particular files. (A "file" is understood to mean a collection of related data found in the mass storage medium.)

Program Storage

Program storage off line is a most useful application of the mass storage facility. Programs can be stored in many ways, including memory images, relocatable programs and source programs. In this utilization the programs and subroutines become part of a software library maintained in mass storage until you need them.

A memory image is one of the simplest forms of program storage. In an inexpensive computer system, the procedure for developing software will start out with some form of system monitor and debugging program which is resident in memory. This is used to load and test a hand assembled or machine assembled application program. Finally, when the program has been debugged satisfactorily, a "snapshot" of main memory can be taken using the audio tape. The routine which performs this function simply copies each byte of memory onto the tape in a fixed sequence starting at a particular address. This process freezes a magnetic picture of the memory on the cassette tape, a picture which can be later recovered by a suitable loader program. In the simplest versions of such a facility, a binary image is recorded on tape; and manual positioning, possibly aided by a tape position counter, is used to locate separate places on a cassette. In more complex versions of the software, facilities can be implemented to specify *names* for the images written on the tape, so that the loader can search the tape automatically for a selected program when it comes time to use the results of your efforts.

A second program storage format mentioned is a relocatable binary format. In this format, the actual image of the program in machine code is copied as in the simpler method. However, the process of creating the mass storage tape also creates various dictionaries of information about the program which allow a relocating loader to move the position of the program in address space. (The simpler memory image method assumes that the program will always be reloaded at exactly the same place from which it was copied.) This relocatable format is common as the output of compilers and assemblers, but it typically requires painstaking detail work to create a relocatable program by hand.

When your system has been expanded to the point where an interpretive language (i.e., BASIC), a self compiler or a self assembler can be run, then program storage can also be done in the form of ASCII character representations prepared using a text editor. This is called a "source" program representation and is directly readable by humans who know the particular computer language involved. (The binary or machine language form of a program which is its memory image is not so easily read by humans.) With such ASCII text representations of programs, some form of language processing program must be used to convert the data into machine language before the program is executed. In practice, such source program representations are typically used for programming and debugging, with the machine language images reserved for use after the program seems to be working acceptably. With source program files as with memory image and relocatable programs, naming is often convenient so that automatic methods can be used to search the tape(s) for the proper file when editing or compiling the program.

Data Storage

Using off line storage for data is a generalization of its use to store programs. The codes used to represent programs are functionally identical to arbitrary data when IO interfaces are considered. The formats used to store data are similar to the formats used for programs in most cases: Data files which are intended to be printed or displayed at some point in time contain ASCII text; data files which have other uses in your system contain arbitrarily encoded bit patterns which depend upon the application of the data.

The source files of programs mentioned previously are simply character text data

Continued on page 95

The COMPLEAT Tape Cassette Interface

Jack Hemenway
151 Tremont St, 8P
Boston MA 02111

The software of a tape cassette interface provides open, data transfer and close operations for both input and output.

Mass storage is one of the most important functions in the small computer system design. Mass storage can typically be used as the medium of a text editor, as the input and output of a full fledged language translator program, and as a means of saving working and debugged software you've created. One of the least expensive ways to accomplish mass storage is the audio cassette storage method.

What is involved in the use of audio cassettes for mass storage? Here's an answer which works quite well in my Motorola 6800 microcomputer system. The COMPLEAT Tape Cassette Interface consists of tape input and output software, the Lancaster speed independent audio interface (see BYTE, September 1975), a Motorola asynchronous communications interface adapter (ACIA), a transmit clock, and the circuitry needed to start and stop the tape recorder's motor under program control. The hardware of the interface is shown in block diagram form in figure 1. The software consists of an open, data transfer and close subroutine for each direction of transfer, input and output. The hardware and software described in this article can be used as the stepping stone to a more complete cassette tape information management system, or it can be used alone whenever a program requires cassette input or output functions.

What Is an ACIA?

The Motorola MC6850 asynchronous communications interface adapter is a specialized version of the familiar universal asynchronous receiver transmitter (UART). The ACIA is designed specifically to interface the Motorola 6800 central processor; however, its use is by no means limited to the 6800. An ACIA can be used conveniently in any computer system with data paths 8 bits or more in width.

The ACIA differs from the conventional UART in the way it is controlled. All control, status and data transfers are made over a single 8 bit bi-directional bus. The integrated circuit contains a control register which may be set by the microprocessor; it is a location in memory address space. The ACIA contains a status register which may be tested by looking at the same location. The ACIA also contains transmitter and receiver data registers which are treated as a memory location via the bus structure and selection logic. In contrast to the UART with its separate input and output data buses, hardwired option selections and 40 pin package, the ACIA design fits into a 24 pin package with several pins left over for use as address selection and modem control functions.

The ACIA options are normally selected by storing a bit string into the control

register when the computer system is first initialized (at power on time) or later when the reset operation is performed manually. However, since the ACIA has the control register, these options can be changed at any time by a program which runs the interface. This capability is used to advantage in the COMPLEAT Tape Cassette Interface: The tape cassette motion is controlled through the RTS line of the ACIA (pin 5, IC1) by setting an appropriate two bit code into the transmitter control bits of the control register (bits 5 and 6); whenever the tape motion is changed (on to off, or off to on), these bits of the control register are redefined.

The ACIA is interfaced to the system data bus either directly, or by means of an appropriate 8 bit bus buffer. The interface is controlled by means of the read write line (RW) and address selection logic. In the hardware of this article, a full address decode is avoided by wiring the chip select lines to appropriate system address bits and using the low order address bit as the register select line (RS, IC1 pin 11). The ACIA has four registers, but only two memory address space locations are required. The apparent inconsistency is resolved by the read write line of the system interface. Two of the internal registers are read only registers (receiver data and status registers), and two of the internal registers are write only registers (transmitter data and control registers). Table 1 shows the system addresses and register access used by the interface of figure 2. (Note that any pair of neighboring locations in memory address space can be used conveniently with appropriate decoding.)

The enable line (E, pin 14 of IC1) is used to synchronize the ACIA status and control changes to the processor, and to condition the ACIA's internal interrupt circuitry. The interrupt request line (IRQ, pin 7 of IC1) is used in systems which employ interrupts to coordinate IO operations. If used, it signals the microprocessor whenever the ACIA is requesting an interrupt. In the simple interface presented here, interrupts are ignored and the software is coordinated using the status register flags of the ACIA.

For a full description of the ACIA control and status registers, the specifications of the Motorola MC6850 ACIA integrated circuit should be consulted. See also pages 3-22 to 3-25 of the Motorola *M6800 Microprocessor Applications Manual*. The software shown in this article makes use of the status register bits for timing and error detection, and sets up the control register for a standard 8 bit asynchronous data format with one start bit, one stop bit, odd parity and a division ratio of 16 for the clocks used with the ACIA.

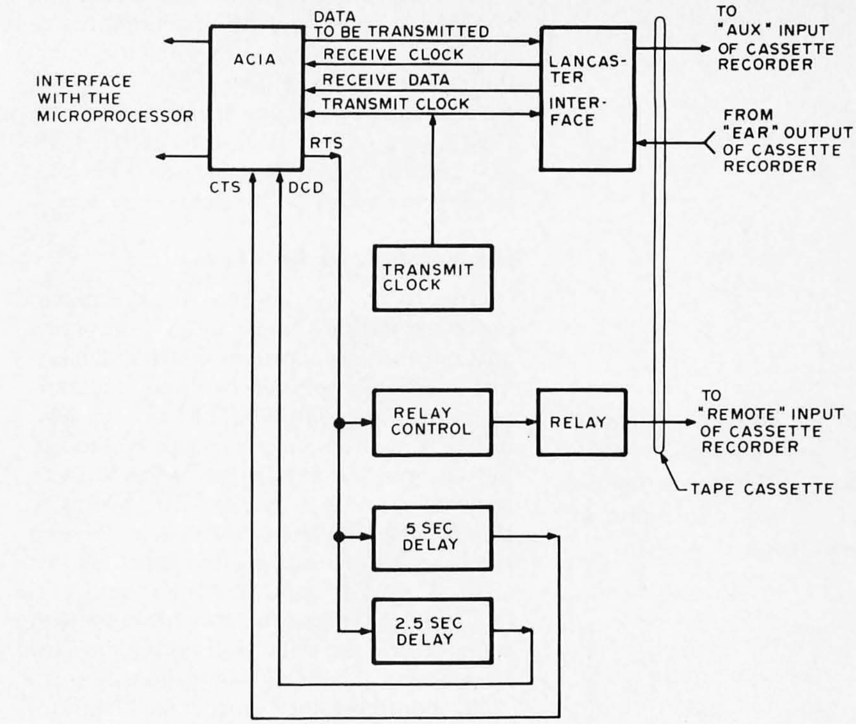


Figure 1: Block diagram of the COMPLEAT Tape Cassette Interface. This illustrates the major elements of the interface; see Don Lancaster's BIT BOFFER article in this issue for details of the hardware of the cassette modem.

On the peripheral side of the ACIA there are three lines which are used to control and test the tape recorder interface. An output line called request to send (RTS, pin 5 of IC1) is used for tape motion control. An input line called clear to send (CTS, pin 24 of IC1) is used for a tape output delay timer, and an input line called data carrier detect (DCD, pin 23 of IC1) is used for a tape input delay timer. Serial data generated by the ACIA is sent to the Lancaster tape interface modem over the transmit data line (TXDATA, pin 6 of IC1), and serial data received from the Lancaster tape interface

An ACIA is Motorola's version of a UART.

Table 1: ACIA addresses. This table shows how the four ACIA registers are referenced, using two memory locations. The secret is that two of the registers are input only, and two of the registers are output only. Thus at each address, the register referenced in the ACIA depends upon whether the CPU is reading data from that address or writing data to that address.

Address	Operation	Symbol	ACIA Register	Typical Code	
8010	read	ACIACTRL	status	LDAA	ACIACTRL
8010	write	ACIACTRL	control	STAA	ACIACTRL
8011	read	ACIADATA	receiver data	LDAA	ACIADATA
8011	write	ACIADATA	transmitter data	STAA	ACIADATA

modem is presented to the receive data line (RXDATA, pin 2 of IC1). During transmission, the data rate is set by the transmitter clock, generated by IC3 and IC4, and during input operations the receiver clock (RXCLK, pin 3 of IC1) is recovered from the tape data by the Lancaster interface, locking the ACIA to the actual tape speed.

Hardware Software Interfaces

The ACIA is controlled by the microprocessor software which views it as the two adjacent memory locations shown in table 1. The interfaces between hardware and software can be controlled by one of two different methods. The interrupt method of IO synchronization relies upon the ACIA to generate an interrupt in the processor through the IRQ line of the system. Because the central processor is interrupted (and its state is saved) only when IO service is required, the processor can be busy with some other task while waiting for the slow IO device to complete its operation.

In this interface, software is synchronized to hardware, using the technique of testing status bits in wait loops.

In contrast, the programmed transfer method employs a wait loop in a program to monitor ACIA status register bits which indicate the progress of the data transfer operations. When the status bits indicate that the ACIA is ready for a transfer to or from the data location, the interface program can then proceed to carry out the transfer. The programmed transfer method is employed in the software of the COMPLEAT Tape Cassette Interface illustrated here, primarily because of its simplicity.

A flip flop toggled by a clock produces an output clock which is a perfect square wave at one half the input frequency.

Reading is accomplished by testing the status register repeatedly until the receiver data ready flag (bit 0 of the status register) is high, indicating the presence of data. When the data is available, the program loads the ACIA data location into an accumulator, an operation which resets the status flag; for input the status register is also tested for the three kinds of error conditions, and a condition code is returned from the input routine in the other accumulator. Similarly, writing is accomplished by testing the status register repeatedly until the transmitter buffer empty flag (bit 1 of the status register) is high, indicating an empty buffer which can receive the output character. The output program then stores a character into the ACIA data location causing the ACIA to begin an output operation and resetting the flag status bit.

Software timing loops and hardware oneshots can accomplish the same goal: delaying execution.

Hardware for COMPLEATness

The circuit of the interface is shown in figure 2, including all details except the Lancaster tape cassette modem circuit. Note that with appropriate clock frequencies, any modem circuit could be used which accepts the asynchronous data format and clock information. (See Don Lancaster's article "Build the BIT BOFFER" in this issue, and "BYTE's Audio Cassette Standards Symposium," page 72 in the February 1976 BYTE.) The modem interface consists of four signals:

TXDATA: This signal is the output data generated by the ACIA at a baud rate equal to the TXCLK frequency divided by the ACIA divide ratio. In this article, division by 16 is used, as set in software by the control codes to the ACIA. This line is shown wired directly from the ACIA, so it can drive the equivalent of one TTL load.

TXCLK: This signal is the output data clock, which is fed to the ACIA and to the tape interface. The Lancaster interface modem uses this signal to synchronously generate the two data frequencies which are recorded on the tape according to TXDATA.

RXCLK: This signal is the input data clock, which is recovered from data by the tape interface. Since this signal is derived from the tape input data, it is locked to any variations in tape speed. Thus the ACIA's input circuitry will not make errors due to differences between the transmitter clock frequency and the variations in tape speed which are called "wow and flutter" in audio tape recorder specifications.

RXDATA: This signal is the input serial data recovered from the Lancaster audio interface modem. Since RXDATA is locked to RXCLK, speed variations of data relative to clock cannot occur.

Transmit Clock

The transmit clock is provided by a 555 oscillator (IC3) followed by a flip flop (7473, IC4) which divides the oscillator frequency by 2. The 555 is wired with components selected for a frequency of 9600 Hz. When the interface is constructed, the potentiometer R1 should be adjusted so that the frequency is 9600 Hz, using a frequency counter or an oscilloscope to make the measurement. For those using oscilloscopes, 9600 Hz is a period of 104.2 μs, or a 5.21 cm trace on a scope set for a 20 μs/cm horizontal time base.

The division by 2 which follows the oscillator is provided by a JK flip flop set up to toggle. This means that both the J and the K inputs are connected to logical one (IC4

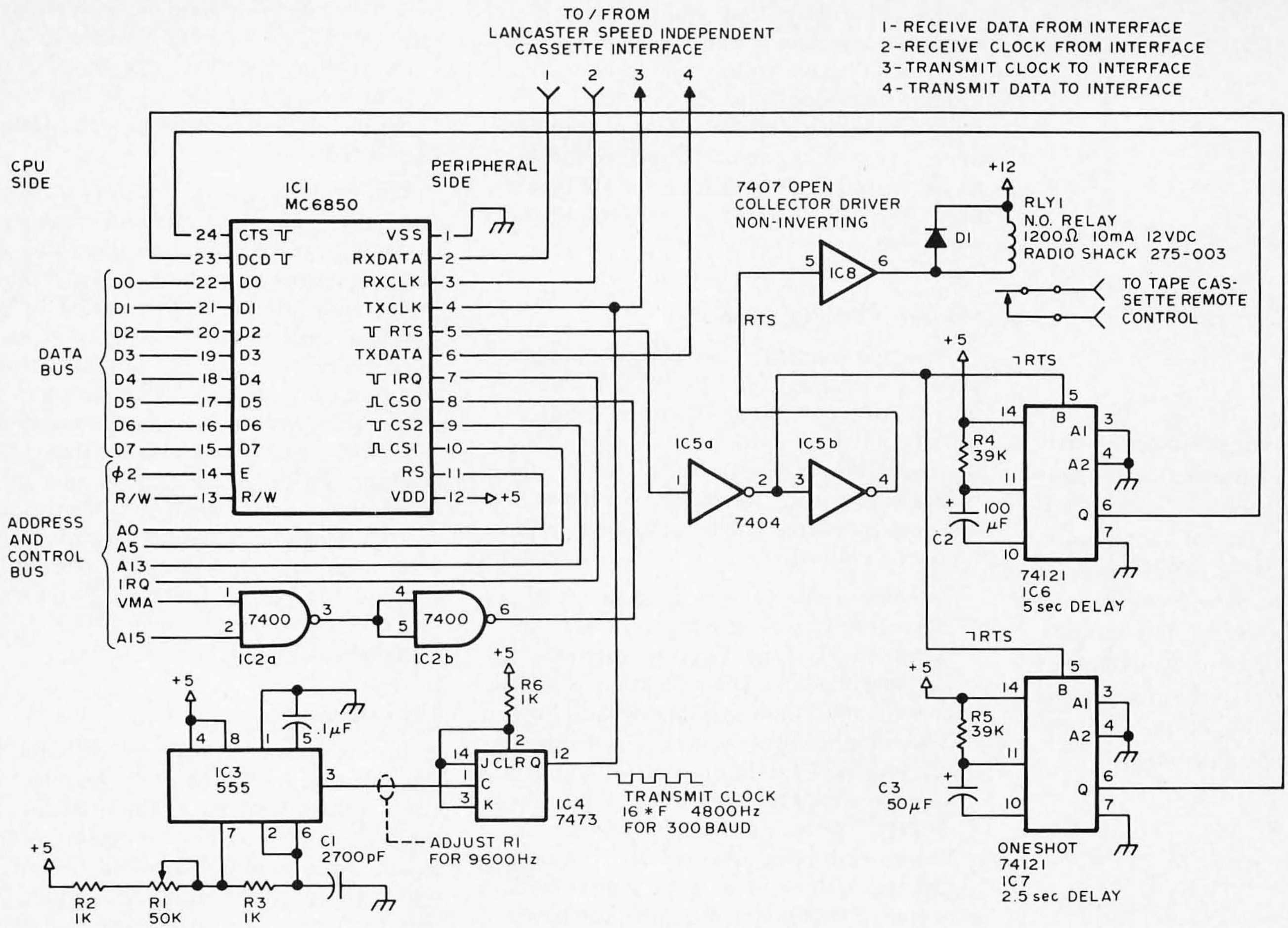


Figure 2: Motorola 6800 ACIA and control circuitry for the COMPLEAT Tape Cassette Interface.

pins 14 and 3). The purpose of the division stage is to produce a perfect square wave clock signal, which is a requirement for the Lancaster cassette interface.

Tape Motion Control

The request to send line (RTS) is used to control the tape recorder's motor, as mentioned earlier. Whenever the ACIA is set up with a control register code for a low value of RTS, the signal presented to the 7407 (IC8) section used as a relay driver is low (after two inversions in IC5). This signal is buffered by the driver, producing a low state at its output (IC8, pin 6) which places 12 volts across the relay coil, closing the contacts and turning on the motor. When RTS is set high, using a different ACIA control code, the input to IC8 is high, so the relay coil has zero volts across it and the relay contacts are open. Note that diode D1 is placed across the relay coil to guard against inductive back EMF which can blow out integrated circuit drivers such as IC8.

Tape Motor Start Delays

Two oneshots are provided in this design in order to give hardware delays of 2.5 and 5.0 seconds following tape motor turn on. The long delay is used prior to output operations so that a long leader at the mark frequency will be recorded. The short delay is used during read operations so that reading will start 2.5 seconds prior to the first actual data byte. Since the asynchronous data format is used, the solid mark tone for about 2.5 seconds will not cause any data to be input; it provides tolerance for manual tape positioning to selected blocks using a tape position counter which is built into many cassette recorders.

The system as designed and illustrated in this article uses hardware to generate the time delays of 2.5 and 5.0 seconds after motor start. It should be noted, however, that the timers IC6 and IC7 could be omitted and replaced by software. To make such a change, the input and output initialization routines would have to be altered to use software timing loops to create the required delay. Examples of such timing loops have appeared in previous issues of BYTE (see "Add A Kluge Harp to Your Computer," October 1975, page 14, and "Can Your Computer Tell Time?", December 1975, page 82). This is an example of a

Always program with commentary – if you want to communicate what you mean to yourself (five years from now) or to your neighbor.

hardware software tradeoff: If you find it easier to program timing loops and you don't mind having an idle computer wasting time in such loops, then omit the hardware timers and use software; if you plan to use interrupts, with the central processor turning to other tasks while waiting for IO operations, then the hardware timers would be preferable.

Software COMPLEATness

No tape cassette interface is complete without software to run it. The software of the COMPLEAT Tape Cassette Interface gives facilities to perform several operations. In order to understand the use of the software provided, the three operations of opening, transferring data and closing a file should be defined:

The larger a block of data on a cassette, the less tape is wasted in "inter record gaps" as the motor starts and stops. At 300 baud, a 4096 byte block can be recorded in 150 seconds, so the 5 to 10 seconds of inter record gap waste less than 7% of the available tape on a cassette.

Opening a file. The first operation in a data transfer to a device such as the COMPLEAT Tape Cassette Interface is opening the file. This operation is minimal for the simple system discussed here: The tape motion is started and a wait loop is entered until the motor start delay is complete. For output, the subroutine T1OTZ performs this operation. For input, the subroutine T1INZ performs this operation. In a more sophisticated software system, opening a file can have a much more general meaning and effect. For the COMPLEAT Tape Cassette Interface, the tape recorder must be set up manually for playback (read operation) or record (write operation) and the tape should be positioned at the beginning or at a location specified by the location counter of the recorder prior to opening the file.

Data Transfer. Once the file is opened, the motor is on; and data transfer can occur for as long as software desires. A data transfer operation is the input or output of one character from the computer. The software of the interface provides an input routine called T1GET which reads the next character into accumulator A with an error condition code in accumulator B. The software of the interface provides an output routine called T1PUT which takes a character from accumulator A and stores it in the ACIA for conversion and output to the cassette modem. Note that the program which calls the data transfer routines must keep track of how many bytes to transfer. One convenient way to do this is to arbitrarily decide to always output a fixed number of bytes, such as 256. Another way to keep track of block size is to decide to send the block length as an 8 (or 16) bit number which is always recorded as the first byte (or first two bytes) of the block. These are by no means the only software data formats possible.

Closing a file. When the software which requires an IO interface completes sending all the data required for one block on the tape, the last step of the IO operation is to close the file. In this simple cassette interface, the file closing operation consists of turning off the motor. (In the case of output, the last character transmission must be completed so the close routine also includes a wait loop.) In the more general case of an information management system, closing a file might include other operations such as recording check sums for error detection. The output file closing routine is T1OSTP, and the input file closing routine is T1ISTP.

The Listings

Listing 1 starts a detailed presentation of the software in the absolute machine language and the symbolic assembly language of the Motorola 6800 microprocessor. (While the interface is shown oriented toward my 6800 system, the listings are given with ample commentary to document function and facilitate conversion to other microprocessors.) Listing 1 contains three statements which set up information used by the assembler. Lines 1 and 2 use the EQU pseudo operation to set the addresses of the labels ACIACTRL and ACIADATA which are used to reference the two memory address space locations associated with the COMPLEAT Tape Cassette Interface hardware. Line 3 is an ORG statement which is used to set the location counter of the assembler to hexadecimal 1000 which will become the starting point of the first routine. While the listings of the COMPLEAT Tape Interface routines in this article show a hexadecimal starting address of 1000, the subroutines can in fact be relocated to any starting address without changing the absolute machine code. If you do choose to relocate the code, however, you should figure out the relocated addresses of the subroutine entry points so that they can be referenced by software which uses these routines.

Listing 2 describes the output initialization routine T1OTZ which is used to prepare for an output operation. When T1OTZ is given control, an assumption is made that the cassette recorder has been manually prepared for recording but with motor

power removed. The ACIA is first reset using the control register code of hexadecimal 5F (lines 6 and 7). Then the ACIA control code for normal operation, hexadecimal 1D, is set up by execution of lines 8 and 9. This turns on the tape recorder motor and triggers the oneshots of IC6 and IC7 in the interface. The output of oneshot IC6 is monitored as bit 4 of the control register (the CTS line into pin 24 of IC1). The initialization routine falls into a loop at lines 10 to 12 until this time delay signal ends and bit 4 of the control register becomes zero. T1OTZ has no parameters and uses the stack to preserve the contents of accumulator B, so that upon return none of the internal registers of the processor have been altered.

Listing 3 describes the output data transfer routine T1PUT. The purpose of this routine is to put the contents of accumulator A into the output data stream. T1PUT first tests the transmitter buffer empty flag of the status register (bit 1) in a wait loop at lines 17 to 19. Then it simply stores the output character of accumulator A into the ACIA data location, which automatically initializes an output operation for that character. T1PUT has one parameter, a character code passed in accumulator A. It preserves the content of accumulator B in the stack.

Listing 4 gives the code for the output close routine, T1OSTP. This routine uses a wait loop at lines 24 to 26 in order to ensure completion of the last ACIA output conversion. Following completion of the last character, T1OSTP loads the ACIA control register with the hexadecimal control code 5D in order to turn off the tape recorder motor. It then returns to the caller. T1OSTP has no parameters. As shown, T1OSTP uses accumulator B as a temporary data storage area but does not preserve its value in the stack; addition of PSHB and PULB operations (after line 23 and before line 29 respectively) could be done to preserve these registers if required.

Listing 5 gives the code of the input open routine, T1ITZ. This routine is identical to T1OTZ in all areas except one: It tests the DCD status line (bit 2 of the control register) instead of the CTS status line. Thus the input initialization routine waits for the 2.5 second delay produced by oneshot IC7.

Listing 6 shows the input data transfer routine, T1GET. This routine is called once for each character of input expected by the program which uses the tape cassette interface. Its first action is to enter a loop (lines 41 to 44) waiting for the receiver data available flag in the status register to become high. When a character is ready and indi-

Listing 1: Global symbol equates and origin of the COMPLEAT Tape Cassette Interface software. This listing sets up the symbolic addresses ACIADATA and ACIACTRL, and sets the location counter to start at hexadecimal 1000. Note that a common statement number sequence is used for all the tape cassette interface software in listings 1 through 8, and that symbols with up to 8 characters (Motorola allows 6) are used in these listings.

```
1  0000  80  10          ACIACTRL   EQU   $8010      set up address of ACIA control
2  0000  80  11          ACIADATA   EQU   $8011          and then ACIA data;
3  1000  10  00                     ORG   $1000      start program at 1000 hexadecimal;
```

Listing 2: Output initialization routine T1OTZ. This subroutine is called after *the tape recorder has been readied manually for a write operation. T1OTZ resets the ACIA and turns on the tape recorder motor, then waits for the end of the output initialization delay. The delay is ended when CTS is found to be zero five seconds after the motor turned on.*

```
4  1000  10  00      T1OTZ     EQU    * this routine initializes and starts output;
5  1000  37                    PSHB                 push B into stack to save it;
6  1001  C6  5F                LDAB   =$5F          ACIACTRL := control code for
7  1003  F7  80  10            STAB   ACIACTRL         master reset, RTS high;
8  1006  C6  1D                LDAB   =$1D          ACIACTRL := control code for
9  1008  F7  80  10            STAB   ACIACTRL         RTS low, normal operation;
10 100B  F6  80  10  T1OTZW    LDAB   ACIACTRL  B := ACIA status register;
11 100E  C5  08                BITB   =$08      test status of CTS (bit 4);
12 1010  26  F9                BNE    T1OTZW    if not ready then keep looping;
13 1012  33                    PULB             else pull B from stack to
14 1013  39                    RTS                  restore it, then return;
```

Listing 3: Character PUT routine T1PUT. This subroutine is called whenever it is desired to write a character on tape. After waiting for a go ahead from the transmitter buffer empty status bit, the routine transfers the contents of accumulator A to the ACIA transmitter buffer register.

```
15 1014  10  14      T1PUT     EQU    * this routine writes one character of output;
16 1014  37                    PSHB             push B into stack to save it;
17 1015  F6  80  10  T1PUTW    LDAB   ACIACTRL  B := ACIA status register;
18 1018  C5  02                BITB   =$02      test status of transmitter (bit 1);
19 101A  27  F9                BEQ    T1PUTW    if not ready then keep looping;
20 101C  B7  80  11            STAA   ACIADATA  else transmit a byte from A;
21 101F  33                    PULB             pull B from stack to restore it;
22 1020  39                    RTS              return to the caller;
```

Listing 4: Output close routine T1OSTP. This subroutine is called following output of a series of characters (a "block"). T1OSTP waits for the completion of the last output operation, then shuts down the tape recorder motor and returns.

```
23 1021  10  21      T1OSTP    EQU    * this routine stops tape after writing a block;
24 1021  F6  80  10            LDAB   ACIACTRL  B := ACIA status register;
25 1024  C5  02                BITB   =$02      is transmitter data register empty?
26 1026  27  F9                BEQ    T1OSTP    if not keep waiting for empty;
27 1028  C6  5D                LDAB   =$5D      else ACIACTRL := control code for
28 102A  F7  80  10            STAB   ACIACTRL      RTS high, motor off;
29 102D  39                    RTS              return to caller;
```

Listing 5: Input initialization routine T1INZ. This subroutine is called after *the tape recorder has been readied manually for a read operation. T1INZ resets the ACIA and turns on the tape recorder motor, then waits for the end of the input initialization delay. The delay is ended when DCD is found to be zero 2.5 seconds after the motor is turned on.*

```
30 102E  10  2E      T1INZ     EQU    * this routine initializes and starts input;
31 102E  36                    PSHA                 push A into stack to save it;
32 102F  86  5F                LDAA   =$5F          ACIACTRL := control code for
33 1031  B7  80  10            STAA   ACIACTRL         master reset, RTS high;
34 1034  86  1D                LDAA   =$1D          ACIACTRL := control code for
35 1036  B7  80  10            STAA   ACIACTRL         RTS low, normal operation;
36 1039  B6  80  10  T1INZW    LDAA   ACIACTRL  B := ACIA status register;
37 103C  85  04                BITA   =$04      is DCD(bit 2) low?
38 103E  26  F9                BNE    T1INZW    if not then keep waiting;
39 1040  32                    PULA             else pull A from stack to
40 1041  39                    RTS                  restore it, then return;
```

Listing 6: Character GET routine T1GET. This subroutine is called whenever it is desired to read a character from tape. After waiting for a go ahead from the receiver data available status bit, the routine transfers the input data to the accumulator A before returning. If errors occur, the error status bits are returned in accumulator B. Note that once the tape is started for input, the processing performed between T1GET calls must on the average be completed **before** *the next character is ready, if an over run error is to be avoided. For a 300 baud transmission rate, this gives 33.33 milliseconds, or 33,000 Motorola 6800 instruction cycles at 1 MHz, assuming that the output routine was called at an IO limited rate.*

```
41 1042 10 42       T1GET    EQU    * this routine reads one character of input;
42 1042 F6 80 10             LDAB   ACIACTRL  B := ACIA status register;
43 1045 C5 01                BITB   =$01      is receiver data ready (bit 0)?
44 1047 27 F9                BEQ    T1GET     if not then keep looping;
45 1049 C5 70                BITB   =$70      are there any errors (bits 4-6)?
46 104B 27 01                BEQ    T1GETR    if not then go read character;
47 104D 39                   RTS              else return with condition in B;
48 104E B6 80 11    T1GETR   LDAA   ACIADATA  A := ACIA data register;
49 1051 5F                   CLRB             B := 0 (clear condition code in B);
50 1052 39                   RTS              return with character in A;
```

Listing 7: Input close routine T1ISTP. This subroutine is called following the last input of a series of characters (a "block"). T1ISTP immediately turns off the motor, since the T1GET routine is assumed to have been executed for the last character prior to T1ISTP. Note that the determination of the length of a block is intentionally omitted from the software of this package.

```
51 1053 10 53       T1ISTP   EQU    * this routine stops tape after reading a block;
52 1053 36                   PSHA             push A into stack to save it;
53 1054 86 5F                LDAA   =$5F      ACIACTRL =: control code for
54 1056 B7 80 10             STAA   ACIACTRL    RTS high and motor off;
55 1059 32                   PULA             pull A from stack to restore it;
56 105A 39                   RTS              return to caller;
```

Listing 8: Test Routines. The programs READ and WRITE are shown in this listing. WRITE should be called first to output 256 bytes of arbitrary data located at hexadecimal addresses 400 to 4FF in memory. Once the block is dumped to tape, the tape cassette can be rewound and set up for a playback operation. Then READ can be called to transfer the data back into the computer where the terminal monitor program (for example, Motorola MIKBUG) can be used to examine the data to verify that the interface restored it properly.

```
57 0100 01 00                ORG    $0100 start programming at location 0100;
58 0100 10 00       T1OTZ    EQU    $1000     here is a set of
59 0100 10 14       T1PUT    EQU    $1014       equates used to
60 0100 10 21       T1OSTP   EQU    $1021       tell the assembler
61 0100 10 2E       T1INZ    EQU    $102E       where the COMPLEAT
62 0100 10 42       T1GET    EQU    $1042       Tape Cassette Interface
63 0100 10 53       T1ISTP   EQU    $1053       is located;

64 0100 01 00       WRITE    EQU    * here begins the output test routine;
65 0100 BD 10 00             JSR    T1OTZ     call the output open routine;
66 0103 CE 04 00             LDX    =$400     X := starting address of block;
67 0106 A6 00       WRITELP  LDAA   0,X       A := memory (X);
68 0108 BD 10 14             JSR    T1PUT     output := A (call the put routine);
69 010B 08                   INX              X := X + 1;
70 010C 8C 05 00             CPX    =$500     is X = last address plus one?
71 010F 26 F5                BNE    WRITELP   if not then reiterate;
72 0111 BD 00 21             JSR    T1OSTP    call the output close routine;
73 0114 39                   RTS              return to monitor;

74 0115 01 15       READ     EQU    * here begins the input test routine;
75 0115 BD 10 2E             JSR    T1ITZ     call the input open routine;
76 0118 CE 04 00             LDX    =$400     X =: starting address of block;
77 011B BD 10 42    READLP   JSR    T1GET     A := input (call get routine);
78 011E C1 00                CMPB   =0        are there errors?
79 0120 26 08                BNE    ENDREAD   if so then stop prematurely;
80 0122 A7 00                STAA   0,X       memory(X) := A;
81 0124 08                   INX              X := X + 1;
82 0125 8C 05 00             CPX    =$500     is X = last address plus one?
83 0128 26 F1                BNE    READLP    if not then reiterate;
84 012A BD 10 53    ENDREAD  JSR    T1ISTP    call the input close routine;
85 012D 39                   RTS              return to monitor;
```

cated by the flag, T1GET transfers the input data from the ACIA receiver data register to the accumulator A at line 48, after verifying that there are no errors in a test at lines 45 to 46. A premature return with the error code in bits 4 to 6 of accumulator B occurs at line 47 if a parity, over run or framing error was detected. The program using the cassette interface is responsible for checking any errors and possibly taking some form of corrective action. If the data had no detected errors, the normal return at line 50 is taken after clearing the error indication code in accumulator B at line 49. T1GET has two parameters which are returned in the CPU accumulators. Accumulator A contains the character which was received (or undefined garbage if in error). Accumulator B contains 0 if there were no errors, and an error condition code in bits 4 to 6 if an error occurred:

- bit 4 is 1 if there was a framing error;
- bit 5 is 1 if there was an over run error;
- bit 6 is 1 if there was a parity error.

Listing 7 completes the tape utility routines with the input close routine T1STP. Since input operations do not have to wait for completion of the last character, simply turning off the cassette motor suffices to complete the input operation. The motor is turned off by storing the hexadecimal code 5F in the ACIA control register.

A Test and Example

In order to illustrate typical use of the COMPLEAT Tape Cassette Interface, a demonstration program was written and is shown in listing 8. This demonstration program has two routines: The routine named WRITE at hexadecimal location 100 should be called from your terminal monitor program (such as Motorola MIKBUG) to copy the contents of hexadecimal memory locations 400 to 4FF onto tape as a single block of characters. (Remember to set up the tape recorder before calling WRITE.) Then the routine named READ at hexadecimal location 115 can be called to read the information back in from tape starting at location 400. (Be sure to rewind the tape and set it up for a playback operation first.)■

REFERENCES

1. Motorola Corporation: *M6800 Systems Reference and Data Sheets, M6800 Microprocessor Applications Manual, M6800 Microprocessor Programming Manual,* all 1975.

2. Lancaster, Don: "Serial Interface," BYTE, September 1975, pp 29–32.

ALTAIR

CONVENTION

Dear Friends,

You are invited to attend the first annual WORLD ALTAIR COMPUTER CONVENTION. This exciting event will be held at the new MITS factory on Saturday and Sunday, March 27-28, in Albuquerque, New Mexico.

Hobby clubs and individual Altair owners have been asked to bring their Altairs to Albuquerque to set up demonstrations. Thousands of dollars worth of Altair equipment will be awarded to the users with the most innovative demonstrations.

Seminars will be conducted by MITS engineers, MITS software writers, and by some of the leading figures in the home computer field.

Since the new MITS factory is located close to the Albuquerque Airport, it will be very convenient for out-of-town guests to attend the convention. The entire Albuquerque Airport Marina Hotel has been reserved for this occasion. This hotel is across the street from the terminal building and just a short walk away from MITS.

Attendence to the convention will be free (not counting travel and hotel costs) and MITS will provide door prizes and at least one free luncheon.

Call or write for hotel and convention reservations. Or for more information:

WACC/MITS
2450 Alamo SE
Albuquerque, NM 87106
phone 505-265-77553 or 262-1951

With the help of the thousands of Altair Users and friends of Altair, it is hoped that this convention will be one of the most exciting computer events in the industry.

Thanks from

MITS
"Creative Electronics"

*Altair is a registered trademark of MITS, Inc.

Magnetic Recording for Computers

William A Manly
Cobaloy Co
626 Great Southwest Pkwy
Arlington TX 76011

Nothing can come anywhere near magnetic recording for low cost per unit of stored information.

Why Magnetic Recording?

Anyone seriously involved with computers, whether he likes it or not, will also be seriously involved with magnetic recording. After one begins working with computers, it doesn't take very long to discover the shocking fact that memory for a computer is going to cost a lot more than the computer itself. A computer requires lots of memory, and professional or amateur, the computer user wants to minimize the cost of his computer setup. A look at figure 1 will immediately tell you why magnetic recording is so important to computer memories: Nothing can come anywhere near it for low cost per unit of stored information. Figure 1 also shows why magnetic recording cannot be used for all types of computer memories: It is the slowest of the memories, which means that it is employed mostly for long term, low usage storage (usually called bulk storage).

All Kinds of Recorders –

Magnetic recorders come in many forms: tape, disk, drum, card, sheet, stripe, roll, cassette, reel, . . . etc. Most of these forms have been used for computer memories in the past, and many are still in use.

And Recording Methods

There are several ways of placing magnetic signals on magnetic media. Among these are those which use the hysteresis loop or the initial magnetization curve, those which use a variation of anhysteretic magnetization, and some methods which use Curie point magnetization. I will go through the first two in detail. The last one involves heating the medium until it is so hot that it is no longer magnetic (it ceases being magnetic at a temperature called the Curie point), then letting it cool in the recording field until it again becomes magnetic. Due to the inconvenience of the temperature cycling, this last method is not important for digital recording. The first method will be covered in the greatest detail, as most recorders designed for digital use employ it. Many of the conclusions drawn will also apply to the second method.

Some other names and subdivisions also apply to the main divisions given above. If we call the first type hysteresis recording, there are two main subdivisions. One is very much like FM radio broadcasting, and is also called frequency modulation recording (sometimes called phase modulation). A single-frequency carrier is recorded on the medium, and its frequency changed according to the information to be stored. Another subdivision is the type used for most digital work. It is called saturation recording. Ideally, the saturation recorded medium has only two states: saturated (magnetized to maximum strength) in one direction, or saturated in the other direction. The information is contained in the transitions, where the direction of saturation is changed. (One older method also used a third state; that of erasure, or zero magnetization.) The second type of recording (anhysteretic magnetization) is also called biased recording. It involves the use of a large amplitude high frequency bias, to which the signal is added. The signal does not modulate (change) the bias in any way. The bias does not return during the signal playback process.

Although the professionals normally use only saturation recording for digital use, computer hobbyists have appropriated recorders intended for other uses, and thus use several types of recording. One is even a type of FM recording using bias to record the carrier. Magnetic recording can also be

classified according to the type of information being recorded, and there is a correlation between the type of information and the type of recording:

Type Of Information	Type Of Recording
Digital professional	Saturation (sometimes FM carrier)
Audio	Biased
Instrumentation	Biased, biased FM carrier, FM carrier
Video	FM carrier
Digital hobbyist	Biased FM carrier, saturation

All of the foregoing seems rather involved, but just remember that the knowledge of a few basics will enable you to sort out almost any recording situation. For instance, all the systems we will discuss involve only a magnetic surface moving with respect to a set of magnetic heads, one of which writes on the surface, and another which reads the information previously written there (if you are an audio enthusiast, forget about the record, playback, and erase heads – those terms are rarely used in digital recording). You are not likely to have an erase head in your system unless you use an audio recorder. Some systems are especially simple, having only one head which both reads and writes. Sometimes the surface moves and the heads are fixed; sometimes the heads move and the surface is fixed; sometimes they both move; but the important thing is the relative head to surface movement.

A Plan of Attack

It isn't very likely that you are interested in becoming an expert on magnetic recording. All that you want is to understand it well enough so you can exercise enough care to prevent its becoming a problem. Knowing this, I'll just present enough of what is called the theory of recording to give you a feel for how it works, then I'll talk a bit of practicalities with suggestions for smooth operation and maintenance. Magnetic recording theory is divided into two parts: Magnetics and geometry. Let's first look at the magnetics.

Blame It All on the Electron!

Almost everyone knows that the electron is a fundamental particle of electricity. It also possesses a magnetic field (electrons always have spin; this spin constitutes an electric current going around in a circle; and anytime an electric current is flowing, it generates a magnetic field). Most materials have their electrons placed in such a way that the magnetic fields all balance out to zero, but there are a few materials which don't. With electron spins paired so that one is spinning clockwise and one counterclockwise, the net field is zero. Of the materials with unpaired electron spins, some are put together in such a fashion that the electrons are coupled together. When this happens, if you manage to turn one spin axis, you have to turn its neighbors as well (the magnetic fields point along the spin

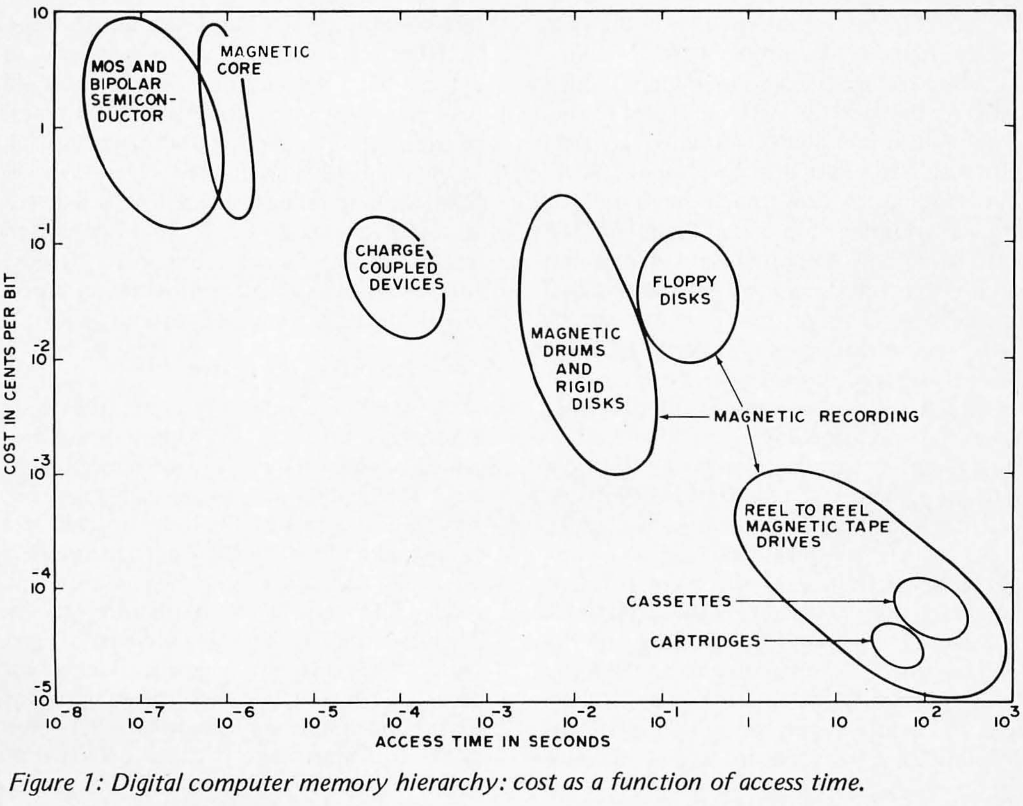


Figure 1: Digital computer memory hierarchy: cost as a function of access time.

axis). Depending on the material, somewhere between a few hundred and a few million of these little fellows will stay coupled together and pointed in the same way all the time. This collection of coupled electron spins is called a domain, and the materials with this type of structure are called ferromagnetic materials.

If a large number of atoms are collected together, there will be two or more domains, whose magnetic fields will not necessarily be pointing in the same direction (though they might). Materials for magnetic recording consist either of domain sized particles separated by non-magnetic material, or they are made of plated material with enough impurities to section the plating into domain sized units. Separating the domains this way allows them to operate nearly independently – a necessity for keeping the information in storage. Such materials are known as "hard" magnetic materials.

Hysteresis, Not Hysteria

A hard ferromagnetic material is characterized by its hysteresis loop. I have a library full of books on hysteresis loops, which have been confusing students for years; but let me see if I can spare you some of the confusion. Suppose we have a material containing a large number of domains whose fields are all pointing in different directions. The fields all cancel out, and the material is said to be demagnetized (note that a single domain cannot be demagnetized). If a very small magnetic field is applied to the material, nothing happens. As the strength of the field is increased, a few of the domains swing their electron spin axes to follow the applied field. As the field strength continues to rise, more and more domains follow the field until finally the last domain responds. After that, no matter how much more field is applied, nothing more can happen. The material is now saturated, and it now has acquired its maximum magnetization, designated M_m. This process is known as the initial magnetization of the material. If we now let the applied field go to zero, a few of the domains decide to desert the pack, but most stay pointing in the same direction. This is known as the remanent condition, with the remanent magnetization designated M_r. Magnetization is given in several units, all of which are measures of how many unpaired electron spins there are per unit volume or unit weight of magnetic material.

Now let's reverse the direction of the field (denoted, for some reason, by the letter "H") and slowly increase the strength from zero. At some point, exactly half of the domains have decided to follow the new field direction, half are still pointed in the other direction; and the result is zero. At this point, the applied field is called the coercive field (sometimes called coercivity or coercive force) of the material, and is indicated by H_c. If the applied field is increased to the former high level, the material again becomes saturated, but in the opposite direction. This cycle can be continued indefinitely, but the material never returns to its erased condition (zero magnetization in the material with zero applied field). If the first direction is chosen to be positive (and the opposite direction negative), we can show a graph of the whole business by plotting magnetization on the Y axis, positive direction up; and the strength of the applied field on the X axis, positive direction to the right. This plot is known as a hysteresis loop, and is shown in figure 2; along with the initial magnetization curve, which is not properly part of the hysteresis loop.

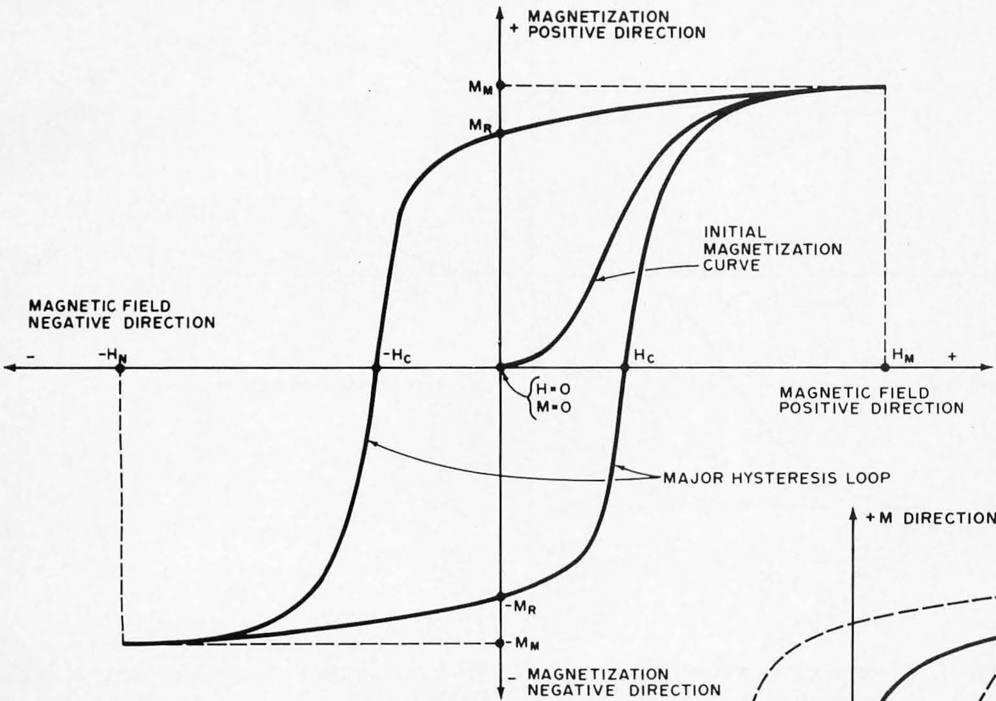
Erasure

If we could limit the discussion to saturation recording, I would have been through with the magnetics right now, but the use of audio recorders has complicated things, so there's a bit more. Suppose we are cycling around the major hysteresis loop we have just described, but start reducing the maximum field a bit each time around. Each time the maximum field is reduced, the loop shrinks in the horizontal direction, and in the vertical direction as well. These smaller loops are hysteresis loops too, but they are called "minor loops." If we continue to cycle, but reduce the maximum field gradually (i.e., go around 10 to 100 times) to zero, the remanence (the magnetization when the field is zero) goes to zero as well. Now we have reduced the magnetic material to the erased condition. It would be well to understand this before going on to the next part, since this cycling and reduction procedure is the basis for biased recording.

Some Recorders Are Biased

Now let's go back to the saturated condition. This time we will apply two fields added together. One is the same large cyclic field we applied in the last paragraph, but the other is a smaller field. The smaller field is applied and held constant. The large field is taken to saturation, then cycled and reduced to zero as in the erasure process. Then the small field is also reduced to zero. Now, the remanent magnetization is *not* zero. In fact, it is larger than one might expect from the application of that small field. This remanence is called anhysteretic

Figure 2: Initial magnetization curve and major hysteresis loop of a hard ferromagnetic material. See text for explanation of abbreviations.



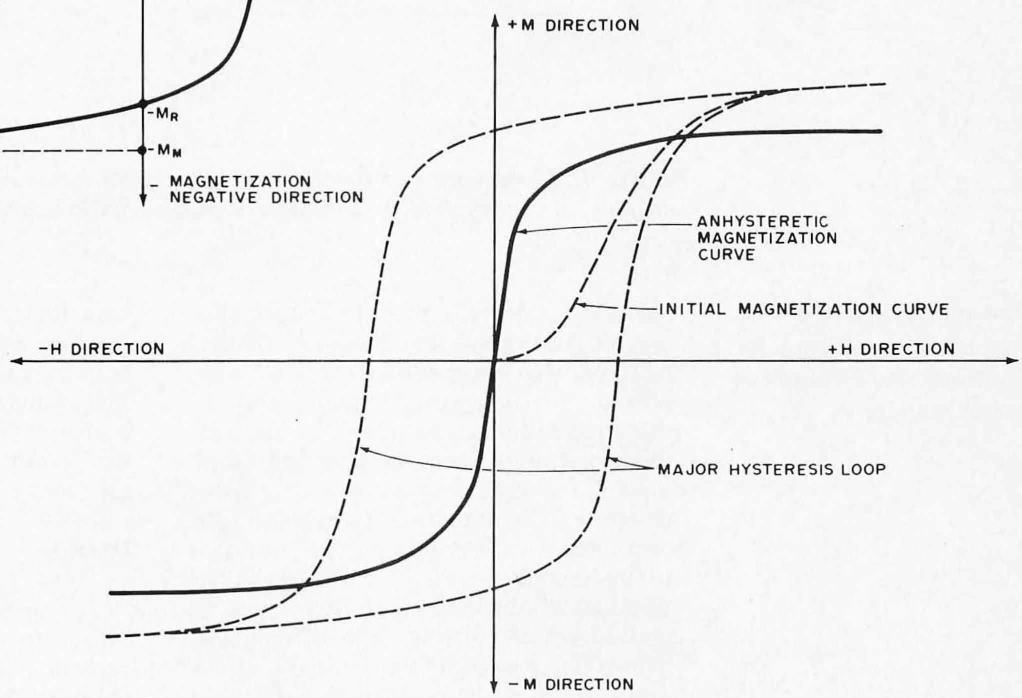


Figure 3: Anhysteretic magnetization curve of a hard ferromagnetic material, compared to the initial magnetization curve and a major hysteresis loop.

remanence. Figure 3 shows a plot of the anhysteretic remanence (solid line) plotted against the small applied field, with the major hysteresis loop shown with dashed lines. This is a transfer curve, which is measured point-by-point, and is not continuous like the hysteresis loop. Note how linear this curve is, and that it is nearly parallel to the sides of the major hysteresis loop. This anhysteretic process is similar to how biased recording works. The large cyclic field is called the bias, and the small DC field is called the "signal."

If a field is applied to an erased medium and then removed, there is some remanent magnetization. If we plot this remanence versus various values of applied field, the curve looks like the solid line in figure 4. Compare it to the linear anhysteretic magnetization curve, which is the dashed curve in figure 4. Its nonlinearity prevents it from being used for audio and other types of recording requiring a linear transfer curve. Note particularly that there is very little remanence until the maximum field is at least as large as H_C. This curve is also a point-by-point curve like the anhysteretic magnetization curve.

An Assist From Euclid

We've covered about all the magnetics you're going to need, so we'll get right into the geometry of the situation. Magnetic

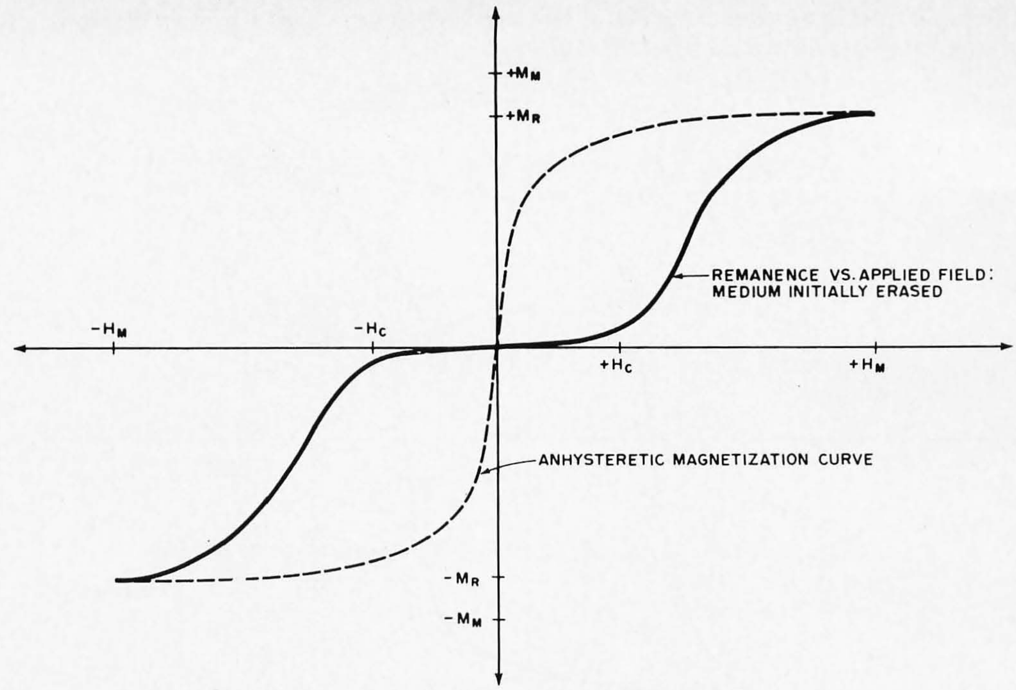


Figure 4: Remanence as a function of applied field for an initially erased hard ferromagnetic medium, as compared to the anhysteretic magnetization curve.

The principal methods of magnetic recording are hysteretic and anhysteretic magnetization.

recording is, fortunately, a two-dimensional process. This means that we can understand most of what we need to know by using only a two-dimensional picture, and the third dimension is thrown in as an afterthought. One of the two important dimensions lies along the recording surface in the direction of head-to-surface movement. The other important dimension is perpendicular to the recording surface, and measures the thickness of the magnetic medium and the head-to-surface spacing. The afterthought dimension measures the magnetic track width. It has to be considered, but it's not nearly so important as the other two.

The particular geometry we'll consider is that of a thick coating. This is the situation with floppy disks, and we'll use them as our primary example. (IBM, who invented the floppies, calls them diskettes. Another term is flexible disks.) The Philips-type cassette is also usually a thick coating (we'll use coating and medium interchangeably) situation, while rigid disks, drums, and most reel-to-reel and cartridge situations are thin media. Thick and thin refer to the ratio of the medium thickness to the write gap length, not to any absolute value of thickness. A thick medium situation exists when that ratio is greater than 0.5, and thin medium situations exist when the ratio is smaller than that. The exact size of the ratio dividing the two cases is a bit arbitrary. Probably not too many computer hobbyists have floppies as yet; but by using a thick medium as an example, we can include characteristics of thin media as a special case. Another reason for picking the floppy is that it uses a type of recording simpler than cassettes use; but, by analyzing it, we can cover all the major principles.

Heads Up!

A ring type head is shown in figure 5a. There are many other types of heads, but this one is well known and widely used, and the principles are analogous for most of the others. Note that this head is balanced: There are similar coils on both sides, and similar gaps on both the top and the bottom. A balanced head has a great resistance to pickup of stray fields, and is used where hum pickup might be a problem. A lot of digital heads are not balanced, and have only one coil, as in figure 5b. Read and write heads usually differ only in detail (gap and track dimensions), or the same head can be used for both functions. Floppy disk drives usually have only one dual purpose head.

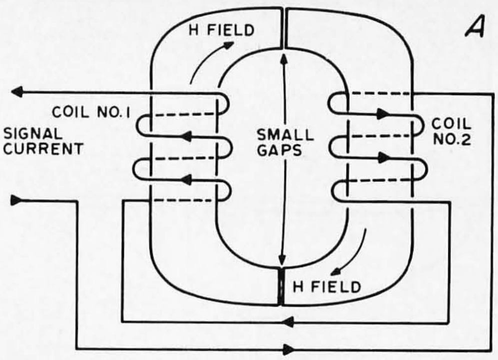
In figure 6, I have blown up the outer edge of the top head gap, and show it contacting the magnetic medium. The actual dimensions of most floppy disk head gap lengths and the coating thicknesses of most floppy disks are about the same: 100 millionths of an inch (100 microinches or 2.54 micrometers).

We're Always Blowing Bubbles

When we create a magnetic field in the write head by passing an electric current through the head coils, the field stays inside the core until it reaches the gap, where it balloons out like a weak spot in an inner tube. Since the head gap is small, the field bubble is confined to a rather small volume. Near the corners of the gap edge, the field rises to a rather high value, even with only a small field in the head core. If the field in the magnetic medium is much higher than the coercivity of the medium, the magnetization of the medium begins to follow the field, and we say that it is being switched. Subsequently, if we allow the field to drop below the coercivity, the magnetization stays pointed in the same direction as the last applied field, and is more or less proportional to the difference between the highest applied field strength and the coercivity (up to the point where the highest applied field strength saturates the material).

Now refer back to the curve "Remanence versus applied field," in figure 4. If we set the write current at a moderate level, some part of the medium is experiencing fields from above saturation (H_m) down to nearly zero. In region A (figure 6) the fields are greater than H_m. In region B the fields are less than H_c and there is little magnetization. The part of the medium in the recording zone (figure 6) will experience a substantial amount of remanence after the field goes to zero (the part of the curve in figure 4 between H_c and H_m). The part of figure 6 labeled "Recording Zone – Low Drive" is a transition region, where some of the material is following the field, and some is not. For most materials, the boundaries are not sharp as shown, but are actually rather fuzzy.

As the medium moves away from the head gap, the part of it which has been in the recording zone has a signal impressed



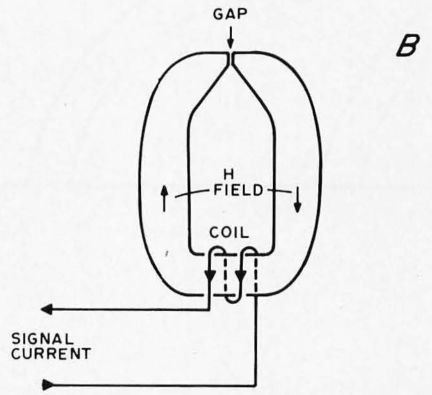


Figure 5: (a) Magnetic ring head, with balanced coils and gaps. (b) Magnetic head with single coil and gap.

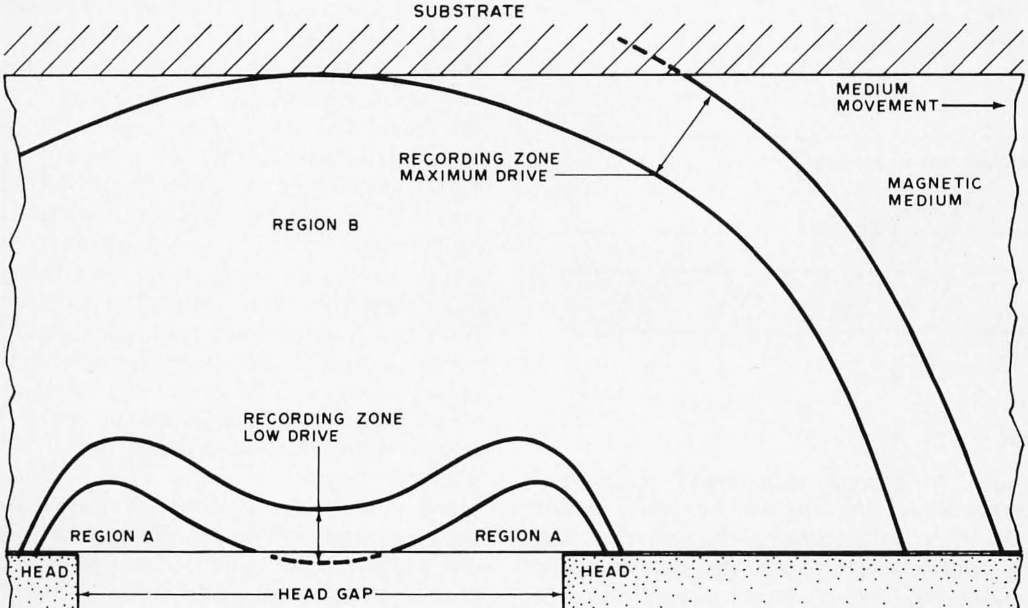


Figure 6: Write head near gap, in contact with magnetic recording medium. Total field near recording zones shown for low drive and maximum output drive.

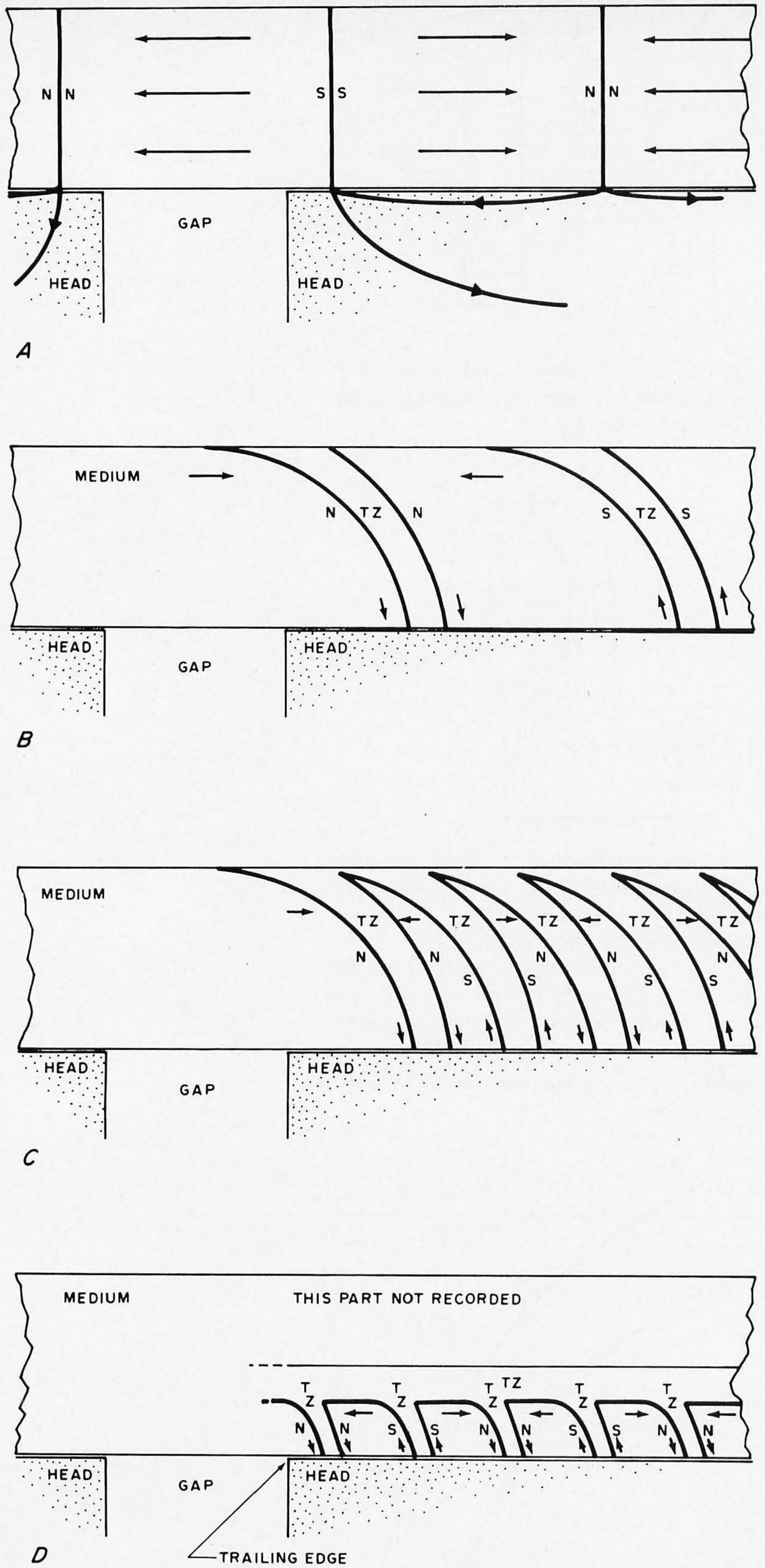


Figure 7: (a) Read head near gap, in contact with ideally magnetized recording medium. (b) Write head near gap, showing two maximum output flux transitions. (c) Same as (b), but with flux transitions extremely crowded. (d) Same as (c), except that write drive has been reduced to relieve crowding effects. Arrows show magnetization direction. N: North-seeking poles; S: South-seeking poles; TZ: Transition Zone.

upon it, while the part farthest away from the head has not seen a field high enough to leave a signal. We can record through the whole coating by increasing the drive through the head coils. With a higher drive current, the transition region is labeled "Recording Zone – Maximum Drive" (figure 6). Note how the width of the transition region has increased; this is a fundamental limitation of the medium and the head.

Things Are in a State of Flux

Figure 7 shows a series of diagrams of the magnetic flux (lines of force) patterns for a reading/writing situation similar to the floppy disk geometry. The flux intensity bounds determining the transition zones are also shown for the writing situation.

Figure 7a shows the head reading, and figures 7b through 7d show the head during a writing sequence. To simplify things in 7a, the ideal recorded flux pattern (which resembles bar magnets laid end-to-end) is shown. Actual flux patterns are similar, but more complicated. Observe that the flux from all the magnetized segments is shorted out by the head, except for the segment across the gap. In that case, the flux threads itself all the way around the head. When the next magnetized region moves into place, the flux will go in the opposite direction. The head coils have an output voltage only when the flux changes from one direction to the other; and the faster it changes, the higher the output voltage will be.

Floppies Are for Real

Now let's look at a real recording situation. The simplest coding is not used by floppy disk machines; but it illustrates all the principles, and is easiest to understand. It is called NRZ1 (Non-Return to Zero, change at 1) recording. The track is divided into small segments all the same length. If the recording is at a bit packing density of 800 bpi (bits per inch), the segments are 1/800 inches long, or 0.00125 inches (32 micrometers). The read electronics are gated so that they only read signals which come shortly before, to shortly after, the dividing line between segments. During this period, if there is a flux transition from saturation in one sense to saturation in the other sense, a pulse will appear in the gate. The presence of a pulse is a one, and the absence of a pulse is a zero. More complicated coding than this is used for floppy disks. One type is called phase modulation. It uses flux transitions between gates so that a positive pulse is a one and a negative pulse is a zero. There are dozens of other coding schemes for digital recording.

Figure 7b shows a head which has just written two maximum drive flux changes on the medium, which is moving from left to right. Several things are of note: (1) the magnetization directions, shown by the arrows, are vertical in some places and horizontal in others; (2) there is a fairly wide transition zone between saturated segments; and (3) the transition zone is spread along the length of the medium. Compare this to the ideal situation shown in figure 7a, which has: (a) all the magnetization in the longitudinal direction; (b) a zero-width transition zone between segments; and (c) the transition zone lying only in the vertical direction. Each of these discrepancies from the ideal case loses some of the signal. There is an optimum value of drive current to get maximum output for any given distance between flux transitions. If the optimum situation is shown in figure 7b, increasing the drive current would make the transition zones more vertical, but the width of the zones would increase so much that the output would go down. If the drive current is decreased, the part of the coating away from the head doesn't get recorded, and this also reduces the output even though the transition zone width decreases.

Long Bars Are Better Than Short Bars

Now look at figure 7c. Either the medium-to-head speed has been slowed, or the frequency of flux changes increased; so that the flux changes come much closer together. We know that the maximum read output would come from what looked like long bar magnets laid end-to-end (as in figure 7a, but with the magnets even longer). The shorter the bar magnets, the less flux goes through the read head and the more goes through the bar magnet itself (this is known as demagnetization). In figure 7c, there is almost as much transition zone as magnet; the magnets are very short and not at all like bars; and the saturation magnetization does not go all the way through the coating. The read output will drop off so much that reducing the drive current, as shown in figure 7d, will actually increase the output again! In figure 7d, the magnets look more like bars, and the transition zones are not such a large percentage of the magnetized part. The recorded volumes do not go all the way through the coating, but the recorded part far from the head in figure 7c was out of phase with the recorded part near the head. It was really subtracting from the signal, so loss of that part actually increases the read output.

One thing is very apparent in 7c: Half the medium is not being used. For short distances between flux transitions, then, a thick medium is a waste. It's even worse than that. The transition zone is partially recorded, and the part farthest away from the head is making a negative contribution to the read signal output. We find that if we decrease the medium thickness so that we get rid of the continuous part of the transition zone (away from the head), we get some increase in output. Decreasing it too much will diminish the output again, so there is an optimum medium thickness for any digital recording situation. Because of the rapid loss of output as the transitions are crowded closer together, transitions are never placed as close together in digital work as in other types of recording. If this crowding is overdone just a tiny amount, some transitions give such a low output that bits are lost: an intolerable situation.

The Cassette Connection

There is a lot in common between digital recording on floppy disks and digital recording on cassettes, cartridges, or other tape media; but there are some differences, too. One difference is that we have been discussing a medium which is isotropic; that is to say, its magnetic characteristics are the same in all directions. This is not true of tapes, as their particles have been oriented during the manufacturing process, so that they record more easily in the direction of head-to-tape motion, and poorly in the other two directions. This means that the longitudinal component of the field is much more effective in recording than the vertical component is. The corresponding figures for oriented media (to 7b, 7c, and 7d) would always have the transition zone going to a point which would be fixed near the trailing edge of the head gap (see figure 7d), and the zone would slant to the left for low write currents and to the right for high write currents. Even with these differences, the conclusions we have already drawn would hold to a large extent. There is some indication that the vertical part of the write head field causes a type of partial erasure of the recording on the surface near the head, when an oriented medium is employed.

Another difference may be that biased recording is used, instead of saturation recording. The situation of oriented media used with biased recording is fully discussed in reference 1. Other types of recording, including frequency or phase modulated carriers, may be used. Teletype signals transmitted over telephone lines or via radio use a frequency shift type of modulation, where one audio frequency is a one and another is a zero. This type signal can be sent directly to an audio tape recorder with good results, except that it tends to be slow.

Magnetic recording theory is divided into two parts: Magnetics and geometry.

Keep It Clean, Fella!

Looking back on what we have learned about reading and writing digital signals on magnetic media, one thing stands out: The distances involved are very small. The period at the end of this sentence is about 0.02 inches (510 micrometers) in diameter. This is huge, compared to these important dimensions in recording systems:

Item	Dimension In:	Inches	Micrometers
Coating thicknesses:	Floppy disk	0.0001	2.5
	Cassette tape	0.0002	5.1
Head gap lengths:	Floppy disk	0.0001	2.5
	Cassette playback	0.00005	1.3

On a floppy disk, a magnetized volume of material on the surface of the coating away from the head is only about 15% as effective as an equally magnetized volume of the coating next to the head; and this is due only to the increased distance from the head. And as we have seen, it's harder for a write head to magnetize the far part of the coating, making things even worse. It follows that a piece of dust, just large enough to see, between the medium and the head can cause a very large loss of output signal. Something only half as large as that period would cause the complete loss of several bytes of information. In a factory making precision tapes or disks, no smoking is allowed in manufacturing areas; hair is kept covered; and special clothing is worn so as not to get anything on the recording surfaces. Even the smoke from cigarettes, pipes, or cigars will build up on heads and recording surfaces and cause eventual signal loss. Ashes cause instant dropouts (total loss of signal). Dust from any source is to be avoided like the plague.

There's also dust and dirt which comes from the medium itself, or its substrate. Floppy disks and tapes are both made out of a long polyester plastic sheet (called a web) which is coated with a special lacquer containing the magnetic material as its pigment. The original web may be from 12 inches (30.5 cm) to 48 inches (122 cm) wide for floppies, or 6 inches (15.24 cm) to 48 inches (122 cm) for tapes. After coating and drying, the web is usually calendered (pressed between heavy rolls). This smooths the surface to a mirrorlike finish, though it was fairly smooth to start with. Tapes are slit out of the web by shearing. Floppies are cut out with a die which also shears the edges. Tapes and floppies are then cleaned by various methods, since the shearing process leaves some debris behind. If the lacquer is formulated properly, and the shearing and cleaning are done with care, normal usage will not generate very much more dust and dirt to cause problems. If manufacturing is done carelessly or the lacquer is poorly formulated or unstable, usage will cause shed (dust), or worse, a gummy build up on the heads. Both these things tend to push the head away from the recorded surface, with a serious loss of output. Even the best of coatings will eventually cause some build up on the heads, and heads should be regularly inspected and cleaned.

Cleaning methods vary, and several ways are effective. If your machine operator's manual makes any recommendations, follow them. There are some special tapes and disks which are run in the machine for cleaning. Several companies have head cleaning materials and solutions on the market. My favorite concoction is half toluene and half isopropyl alcohol; but it has to be used with care, since the toluene dissolves some plastics and media coating lacquers. Straight isopropyl alcohol does a fair job, and is available in any drugstore. Apply the cleaner to the heads (and guides of a tape machine) with cotton tipped sticks. The ones made especially for cleaning heads are best, since their sticks are stiff, but you can also use the ones made for cleaning and oiling babies. Clean until the coating color is removed, or until the cotton swab comes away clean.

Professional installations sometimes have special machines to clean and recheck their media, but this is not usually within the budget of the individual. Cleaning of tapes is often accomplished by running them across a woven, lightly oiled, soft paper wipe which is moved slowly away from the point of contact. Tapes and disks can also be cleaned in an ultrasonic bath with an air squeegee. All methods require relatively complicated machinery, making cleaning impractical except for the largest installations. There are some companies which make a business of cleaning and re-certifying media. I recommend retiring from digital use any dirty media, and substituting new.

When buying tapes and cassettes, get the best quality you can buy. This is no place to save money, as it is always at the expense of lost bits. Tapes especially made for digital use are a good buy (floppies are always made for digital use). If you can't get these, use the top line of a well known brand of audio tape. Even this is second choice, since audio tapes, even good ones, may have some bumps on the surface which cause dropouts. The loss of five cycles of that high violin in "Scheherazade" will cause only a tiny gap which you won't hear, and you can lose a whole percent or so of "Rites of Spring" and

never know it; but the loss of just a bit or two of a digital sequence can cause nothing but garbage to issue from your computer.

Making Your Media Comfortable

About 15 years ago, some people at Southwest Research Institute, with grant money from the Rockefeller Foundation, made a monumental study for the Library of Congress on storage of sound recordings (reference 2). Part of their study was concerned with magnetic tape. Not very much can be added to their findings today. Boiled down, we can almost put their findings into one sentence: If people are comfortable in an environment, tapes can be safely stored there for long periods of time with little degradation. I say almost, because there are a couple of things to add to this. One is that, other than the earth's field, no other magnetic fields should be present if information is contained on the media. Permanent magnets, wiring carrying heavy currents, power transformers, and magnetic erasers or degaussers should be kept away from the media. For most of these things, three feet (one meter) is a good rule of thumb for distance. Don't get carried away and worry about such things as shielding from the earth's field, protecting from lightning or static electricity, guarding against radiation from radio transmitters or radar sets, or storing a hundred feet away from any electric wiring. Trouble from magnetic fields, though it can occur, is rare. The other added condition is that all media should be stored under low mechanical stress. Tapes and cassettes should be wound properly from a regular run, not a fast wind. Floppies should be stored flat, with no weight piled on top. If supported so that they don't buckle, they can be stored on edge. Never remove them from the envelope if you want to use them again. Avoid large temperature or humidity changes.

Summary

What I have tried to do is give you first an overview of digital magnetic recording so that maintenance and setup instructions for your machine will make sense to you. I haven't given specific directions for maintenance or setup, because each machine is a little different. Knowing how the information is contained on the medium is also of importance to understanding why cleanliness and good storage conditions are so important to safe storage. Lastly, I collected together several guidelines for cleanliness and storage which you probably won't find in the instruction manual for your machine. I hope that all this helps you to pack away your bits for easy retrieval. Once these principles become second nature to you, your large-scale storage problems should fade into the woodwork, and you can then apply your troubleshooting talents elsewhere. ■

BIBLIOGRAPHY

1. "A Primer on Choosing Tape," William A. Manly, *Audio*, Volume 58, Number 9, September 1974, pages 34-46.

2. "Preservation and Storage of Sound Recordings," A.G. Pickett and M.M. Lemcoe, Library of Congress, Washington: 1959. Superintendent of Documents, Washington DC.

GLOSSARY OF MAGNETIC RECORDING TERMS

Anhysteretic magnetization: The magnetization remaining in a ferromagnetic material after applying a constant field H_{fixed}, superimposing on it a field varying continually from $+H_{cycled}$ to $-H_{cycled}$ (which is initially large enough in amplitude to cause practical saturation in each direction, then reducing the amplitude of H_{cycled} to zero as the cycling continues).

Biased recording: Magnetic recording done by adding the signal field to be recorded, a high frequency, large amplitude field called the *bias*. The purpose of the bias is to linearize the recording process.

Bulk storage: Supplemental storage of large volume capacity. Also called external storage, secondary storage or mass storage.

Coercive field: The applied magnetic field in a given direction, necessary to reduce the remanent magnetization of a ferromagnetic material to zero, after the application of a saturating field in the opposite direction.

Curie point magnetization: Magnetization of a ferromagnetic material, acquired by applying a field, heating the magnetic material until its ferromagnetism disappears (the "Curie point"), then cooling the material while still in the field.

Demagnetized: The condition of a ferromagnetic material when the directions of magnetization of all its domains have been randomized, so that there is no external field coming from the material.

Domain: A small volume of a ferromagnetic material in which the atoms are always magnetically aligned in the same direction. The magnetic direction of a domain may be changed, but it may not be demagnetized so long as the material is ferromagnetic.

Electron: A non-nuclear part of an atom; the smallest particle of (negative) electricity. An electron is regarded by physicists as a fuzzy ball of negative electricity which has a "spin" characteristic.

→

Erasure: The process by which a bulk magnetized ferromagnetic material is placed in a bulk demagnetized condition.

Ferromagnetic: A ferromagnetic material is spontaneously magnetized into an assemblage of tiny permanent magnets called domains. A ferromagnetic material can be demagnetized only in a bulk sense, and only when it is of a large enough physical size to contain many domains.

Frequency modulation: The changing of a carrier wave's frequency in accordance with the signal being transmitted.

Hysteresis loop: A closed curve obtained by plotting magnetization for ordinates ("y" direction) and applied magnetic field for abscissa ("x" direction) as the material passes through a complete cycle between definite limits of applied magnetic field.

Hysteretic magnetization (or **hysteresis** magnetization): Magnetization in a ferromagnetic material acquired by the cyclic application of a single applied magnetic field; magnetization at some point on a hysteresis loop.

Initial magnetization curve: The plot of the magnetization for ordinates and the applied field for abscissa of an initially bulk demagnetized ferromagnetic material, as the applied field has its strength increased from zero to some high value.

Isotropic: An isotropic material has some property the same in all directions. This word must be modified by some adverb describing the property, such as "magnetically isotropic."

Magnetic direction: A vector on a permanent magnet pointing from the south-seeking pole to the north-seeking pole; for a magnetic field, the vector starts at the north-seeking pole of a magnet and goes toward the south-seeking pole.

Magnetization: The number of elemental magnetic dipoles per unit volume of magnetic material. A single, isolated, spinning electron can be taken as the elemental magnetic dipole. All other units of magnetization are based on this.

Remanent magnetization: The particular value of magnetization on a hysteresis loop when the applied field is zero; the bulk magnetization of a ferromagnetic material when there is no applied field.

Saturation magnetization: The magnetization of a ferromagnetic material when the applied field is so large that all the domains have their magnetic directions aligned with the applied field.

Spin: A representation of the rotation of an atomic or sub-atomic particle. Spin is a vector pointing along the direction of the axis of rotation.

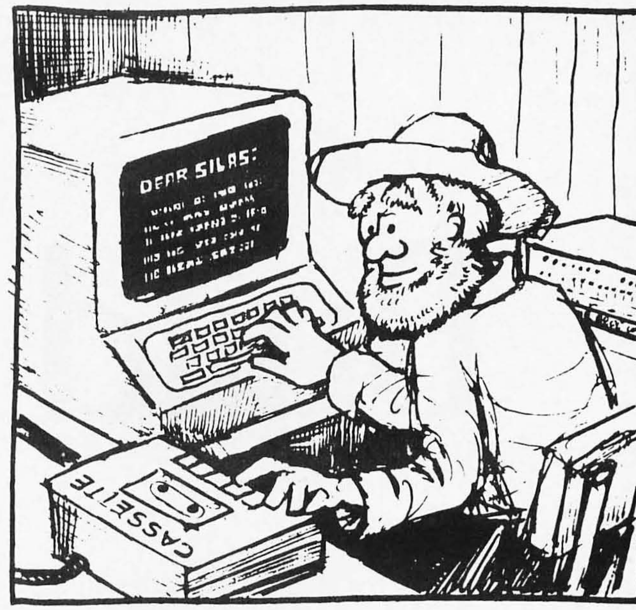
Thick coating: A relative term referring to a magnetic coating or layer such that its thickness is greater than about half the length of the write head gap.

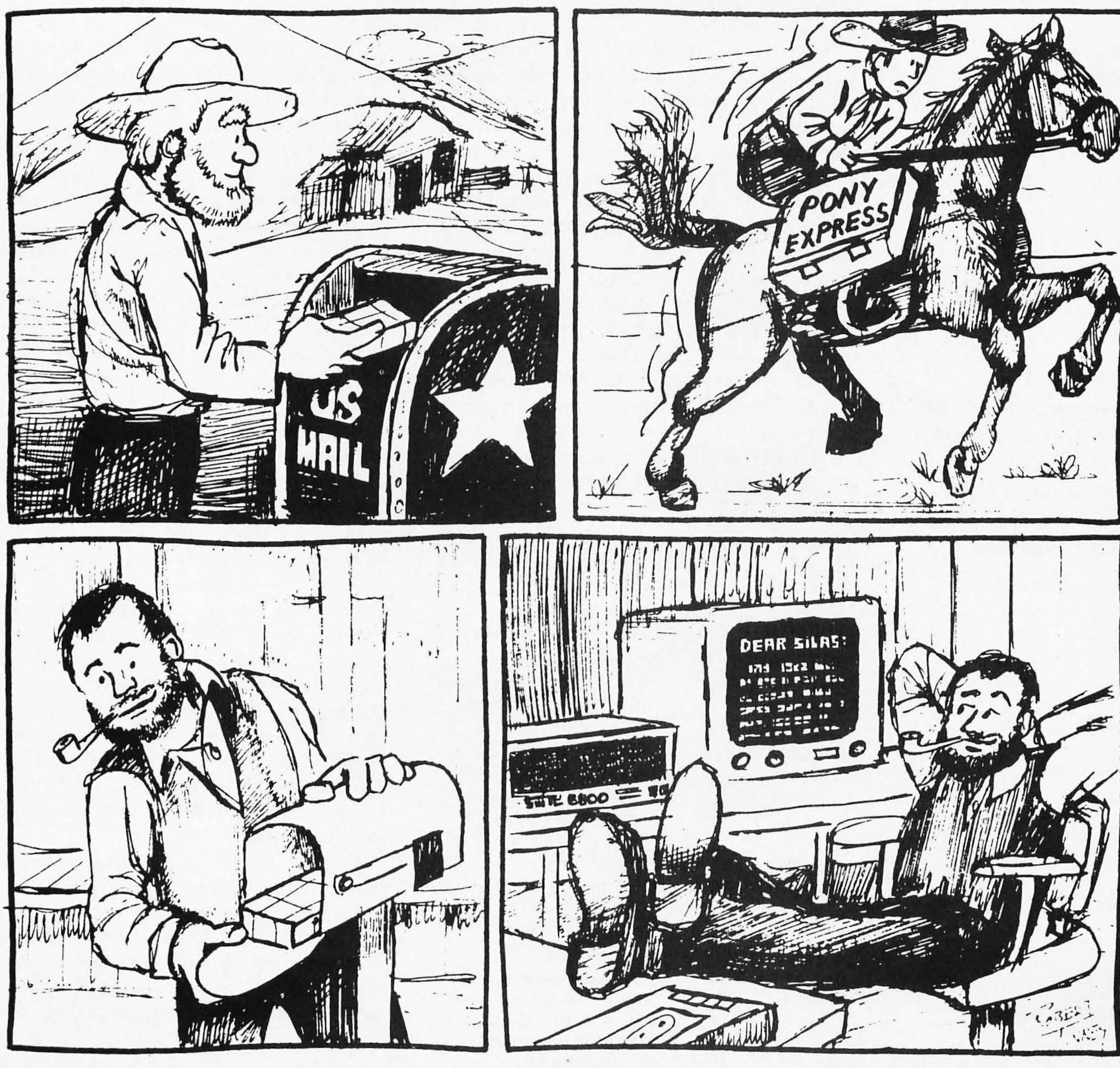
Thin coating: A relative term referring to a magnetic coating or layer such that its thickness is less than about half the length of the write head gap.

Technology Update

BYTE PRESENTS THE LATEST IN INTERCOMPUTER DATA COMMUNICATIONS METHODS...

IDEA: A.N. ONYMOUS   ART: ROBERT TINNEY





Build the BIT

Don Lancaster
Synergetics

The format of the BIT BOFFER is compatible with the provisional audio cassette standard described on page 72 of BYTE's February 1976 issue.

The BIT BOFFER is a low cost way to store and retrieve digital data onto and off an ordinary cassette tape, using stock audio tape recorders. You can also use it to exchange data, recording on one machine and playing back on a second. The BIT BOFFER is totally software independent, meaning that it can be used with or without a microprocessor system. All that's needed to control it is a simple "data ready" command to send something on to the recorder; you also get a "data available" command when the recorder produces an output. The BIT BOFFER can be connected to any serial data port of a microprocessor or other system.

The BIT BOFFER is highly speed tolerant, which lets you record on one machine and play back on a second. Better yet, it lets user groups or software sources provide multiple copies of programs and data simply and at very low cost.

Table 1: Advantages of the BIT BOFFER system.

***SPEED TOLERANT**	Up to ±25% speed variation without adjustment.
***SOFTWARE INDEPENDENT**	CPU or microprocessor not needed for operation. Runs with simple "send" and "ready" commands.
***IO COMPATIBLE**	The BIT BOFFER modem works directly with existing UART or ACIA. Coding compatible with TTY, 103 modems, TVTs and RS232C.
***NON CRITICAL**	One adjustment in system. No preamble or block limits. Adapts to wide range of baud rates.
***CHEAP AND SIMPLE**	Single sided compact PC card circuit costs less than $6 in quantity. Can be redesigned even lower.

The BIT BOFFER is very much compatible with standard serial interface using UARTs or microprocessor serial interface chips. It works with any serial interface that uses 16X clocks and has separate clocks for receive and transmit.

While the error rate of the BIT BOFFER system is more than adequate for hobbyist quality use, software techniques can optionally be added to the basic system to "harden" the error rate for fully professional, quality data storage and retrieval. The format of the BIT BOFFER is compatible with the provisional audio cassette standard described on page 72 of BYTE's February issue. Some characteristics of the BIT BOFFER system are summarized in table 1.

You can use the BIT BOFFER with almost any cassette recorder, although a medium quality unit with an auxiliary input and an automatic level control is recommended *for recording*. Just about any machine will play back a properly recorded tape. You do have to use a reasonable quality tape, free of dropouts and other defects. You also should "certify" the tape for dropouts, but this is only a commom sense precaution.

Cost of the system? Several vendors are now offering versions of this interface; the version we'll show you here uses six (optionally seven) stock CMOS integrated circuits with a typical cost of 50¢ each or so. The circuit fits a small single sided PC board and uses very little current from a single 5 volt supply.

BOFFER

How It Works

Most audio recorders are very fussy about what they are willing to accept as input signals. The cheaper the recorders, the fussier they get.

If the automatic level control of the recorder is going to help us rather than fight us, we want to record a constant amplitude signal of some sort.

What the recording head likes to work with is even more restrictive. For instance, figure 1 shows us the response of an *ideal* audio recording head. Over the useful recording range, the sensitivity doubles with each doubling of frequency. Go too low in frequency and the system noise level interferes with recording. Go too high and a gap resonance cancellation is caused when the physical length of the signal recorded on the tape exactly equals the gap width.

Doubling response as you double frequency is the same as mathematical differentiation. You respond to the *changes* or the *slope* of what you are trying to record, and not to the value of the signal. Note that this slope taking process happens twice, once while recording and once while playing back.

So, let's take some derivatives. The first derivative of a sine function is a cosine function, and the second derivative is a sine function, again of the same frequency. Put in a continuous sine wave and after the second derivative you get back a continuous sine wave, along with a phase reversal or two. But sine waves often require expensive hardware.

Can we use square waves? Put in a square wave and you record an impulse on the leading and trailing edges. Play this back and you get a *double* impulse, one pair at the leading edge and one at the trailing edge of the original square wave. There is no way to get a square wave back for a square wave in.

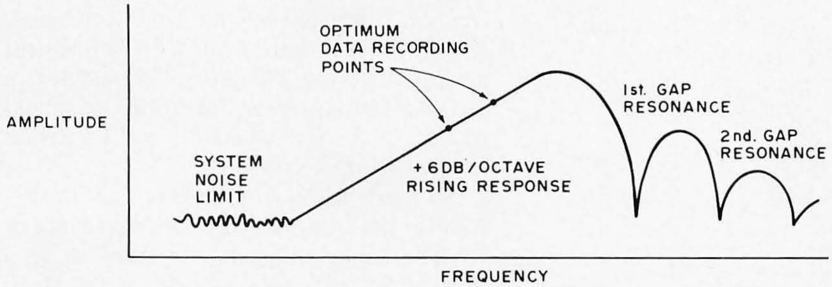


Figure 1: The response curve of an ideal audio recording head.

Now, the computer data recording people pull some tricks to simulate the recording of square waves. They use special heads and levels to saturate the tape in one direction or another. On playback, they sense the flux reversals with sense amplifiers and then use the amplified impulses to set and reset a flip flop, thus regenerating a square wave. Thus, saturating the tape eliminates one of the differentiations, and the set reset flip flop gives us an integration that cancels the second differentiation.

But this is vastly different from an audio cassette head working with linear, non-saturating tape signals. The audio cassette people attempt to compensate for the recording slopes, both on record and playback, to make it more or less flat over the audio spectrum. This compensation doesn't affect sine waves much, but with a square wave, it spreads out the harmonic energy and generally makes a mess of it. This is called the group delay distortion problem, and is caused by different response to the

The BIT BOFFER's data format is a "self clocking" one in which the receiving circuitry is locked to the actual data.

The absolute optimum signal we can record on an unmodified low cost audio cassette recorder is a constant amplitude zero offset sine wave whose frequency is changed only when the sine wave goes through zero.

harmonics of the square wave than to the fundamental frequency. Also, much of the compensation is done by the least expensive method and simply rings badly when driven by a square wave. While it is possible to record square waves on a quality machine, if you want reasonable performance for any machine, the *only* signal you should consider using is a sine wave.

Further, if we turn our sine wave on or off at the wrong time, we introduce a transient that's the same thing as the leading edge of a square wave. This causes considerable distortion for us. We can beat this by using a transient of zero amplitude, which means that the only time you can change the sine wave is as it goes through zero.

From all this (plus a mountain of lost time and painful testing), we can conclude that the absolute optimum signal we can record on an unmodified low cost audio cassette recorder is a *constant amplitude, zero offset sine wave whose frequency is changed only when the sine wave goes through zero*. Better yet, as a refinement anticipating the "coasting" effect of a recorder's group delay distortion, we should change frequency coherently but somewhat *before* each zero crossing.

So much for making the recorder happy. We also have to make our serial interface or UART happy. A serial interface using a single clock gets very unhappy if the transmitted and received speeds differ by more than a percent or two. On reception it starts outputting garbage since the bits at the end of the word get ahead of or behind the "ideal" positions. A high quality, line operated recorder can usually hold its speed variation to within a percent or two, but inexpensive battery operated units cannot. This speed variation is enhanced if we are recording on one machine and playing back on another.

There's a simple way out of this bind. Put a signal on the tape that not only tells you ones and zeros but also *how fast* the tape is going. Use the recovered speed information

Table 2: Tone standards for BIT BOFFER system.

110 baud:
1 = 16 half cycles of 880 Hz
0 = 8 half cycles of 440 Hz

300 baud: *
1 = 16 half cycles of 2400 Hz
0 = 8 half cycles of 1200 Hz

600 baud:
1 = 16 half cycles of 4800 Hz
0 = 8 half cycles of 2400 Hz

***This is the provisional standard resulting from the BYTE symposium.**

to speed up or slow down the UART receiver to match the data rate. The result is called a "self clocking" recording method.

Table 2 shows us some standards for a set of signals that meet the specification of constant amplitude sine waves or *half sine* waves that are zero crossing switchable and still give us the ability to extract self clocking digital information off a single audio cassette track.

To record a one, we record 16 half sine waves of a frequency that is eight times the data rate. For a zero, we record eight half sine waves of a frequency that is four times the data rate. The half sine waves are switched just before their zero crossings and always are phased for a continuous waveform through zero. Timing is controlled by a reference clock at 64 times the data rate. These are often already available in minicomputer systems. The reference clock (64 times data rate) is divided by four to run the clock input of the UART's transmitter at 16 times the data rate. It is selectively divided as needed to synthesize sine waves for recording.

During playback, these signals are amplitude limited to minimize tape variations and interference from bias, hum, and other noise. One clock pulse is reconstructed from each zero crossing. The distance between zero crossings is also measured by a retriggerable monostable circuit. If the distance is too great to be a one, a zero output is provided on the data line, and a new clock pulse is thrown in for the UART receiver circuitry. With this design, you automatically get 16 clock pulses out for each one and 16 clock pulses for each zero. In the zero case, the clock pulse spacing varies somewhat as the BIT BOFFER tracks the tape speed. But this is what makes the scheme work, since the UART receiver doesn't care and will still operate properly.

Figure 2 shows the word and timing formats for the BIT BOFFER system. A typical system might use ASCII characters sent in the Universal Asynchronous Data format that consists of a start bit, the ASCII code starting with the least significant bit, a parity bit, and two stop bits. The width of each bit is set by the data rate.

If we're working with 8 bit binary data instead of ASCII characters, we send a start bit, eight bits of data beginning with the least significant bit, an optional parity bit, and then two stop bits.

Using the asynchronous format, words or characters can go on the tape with any spacing between them. For the most efficient use of tape storage, large blocks are recommended. For example, cycling once

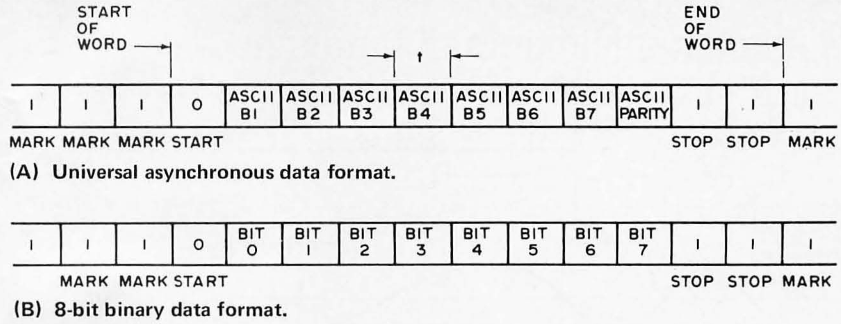
through a typical television typewriter system's memory while driving a UART output (or reading input) would provide 512 character blocks corresponding to the 32 x 16 character display. With a microcomputer system, block sizes can be chosen arbitrarily as a software convention. However, the overhead of starting and stopping the tape can be minimized with larger block sizes. The only preamble or ending needed on the character blocks is some "mark time" or a string of digital ones, similar to the initial and final rubouts on a paper tape system.

Figure 2c also shows the timing standards for recording and playback for the various data rates.

One problem you may have to allow for in your system is *overspeeding*. Overspeeding is caused by recording on a slow machine and playing back on a fast one. The BIT BOFFER can easily take care of this, but the system which is ultimately doing the receiving and using the data has to handle the faster rate. For instance, on a TVT with a single frame update, 60 characters per second is the upper limit on entry. With a Teletype, 10 characters per second is usually tops. To beat this problem when interfacing fixed speed systems, you simply hold the character rate down a little when recording, say to 75 or 80 percent of capacity. This is done by having the data source wait several tens of milliseconds between characters. If the BIT BOFFER happens to speed things up a little, the maximum rate is still under what your system can accept.

Note that the overspeeding problem only exists for systems which assume a fixed input data rate from the interface. In the typical microcomputer system, use of the interrupt structure or status bits from the interface can synchronize a program to the actual received data rate. Since the program is then synchronized to the received data bits (which are in turn synchronized to the actual tape speed), the combined microcomputer and BIT BOFFER interface becomes speed independent within the BIT BOFFER's range of ±25% variation from nominal.

A related problem that's also easy for you to fix is that some of your peripherals may need absolutely constant data rates. This is particularly true of a Teletype, and the playback rate of the BIT BOFFER may be out of tolerance. To get around this, you simply connect your UART's parallel receiver outputs back to its own parallel transmitter inputs and retransmit at the correct rate. This is particularly easy to do with ACIA type microprocessor interface devices with bi-directional parallel IO lines.



(A) Universal asynchronous data format.

(B) 8-bit binary data format.

BASE RATE	110 BAUD	300 BAUD	600 BAUD
Transmit time "t"	9.09 msec ±1%	3.33 msec ±1%	1.67 msec ±1%
Receive time (without adjustment)	9.09 msec ±20%	3.33 msec ±20%	1.67 msec ±20%
Word time	100 msec	36.67 msec	18.33 msec
Recommended max word rate (allows for overplay)	7.5 char/ second	22 char/ second	44 char/ second

(C) Timing standards.

Figure 2: Data format and time standards for several data rates. The BYTE provisional audio cassette standard utilizes the 300 baud data rate; communication with a Teletype requires a 110 baud data rate.

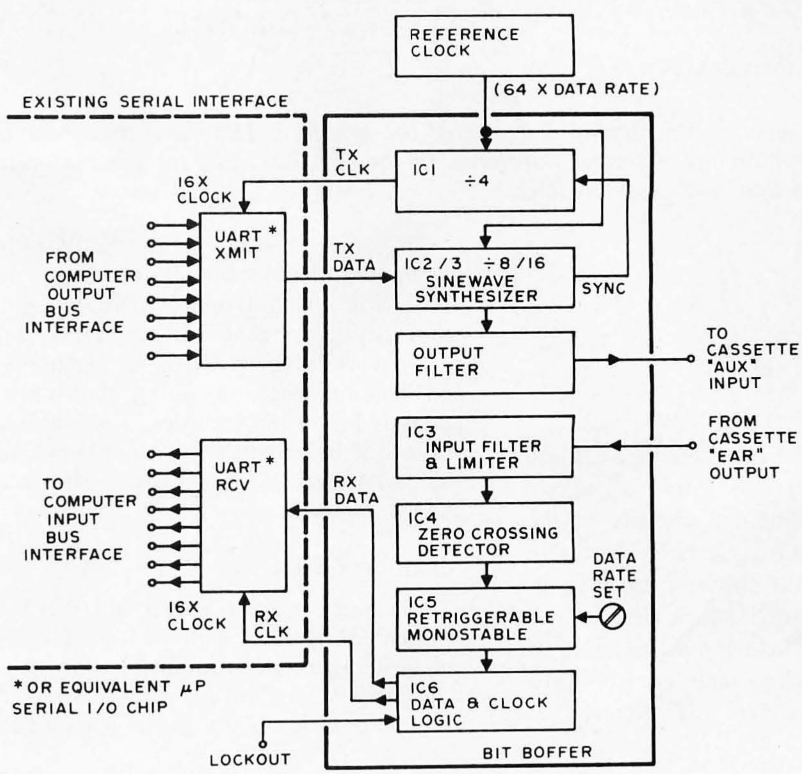


Figure 3: Block diagram of the BIT BOFFER. The integration of BIT BOFFER into a microcomputer system requires a UART or other existing serial IO interface. This is shown at the left in the block diagram. The connections to the recorder are shown at right. The BIT BOFFER proper is enclosed in the box in the middle.

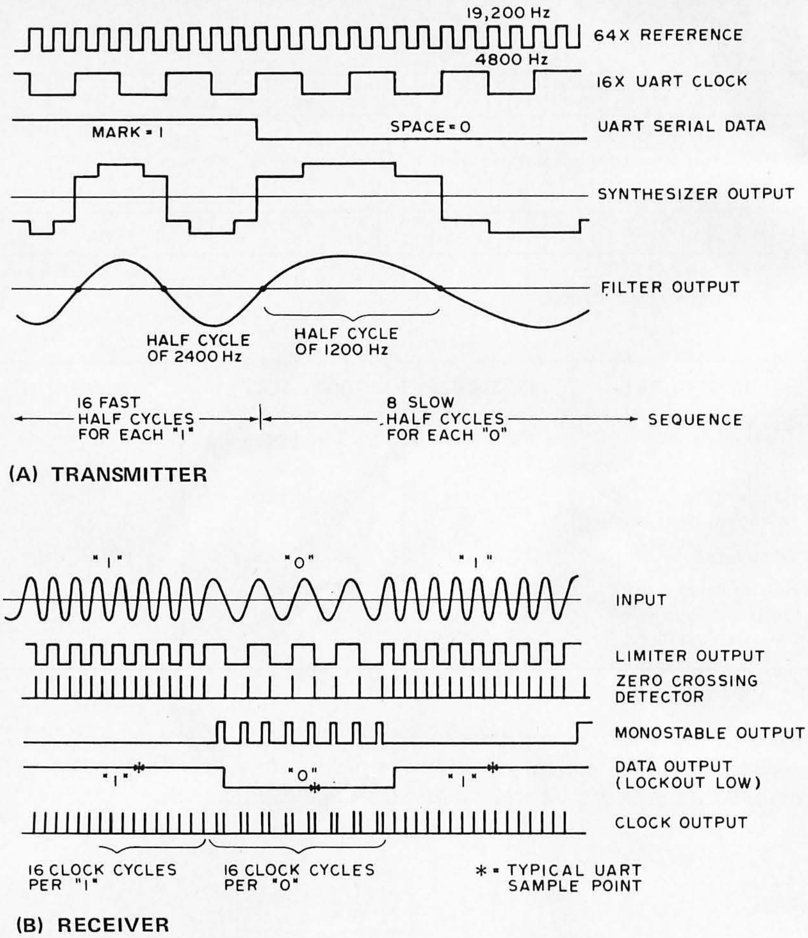


Figure 4: Key timing diagrams of the interface. These diagrams show the relationship of several waveforms in the BIT BOFFER: (a) The transmitter section and (b) the receiver.

The complete tape cassette interface for a microcomputer consists of the BIT BOFFER hardware plus software to drive it. Turn to Jack Hemenway's article for an example of some tape control software for this interface.

An earlier version of the BIT BOFFER system was described in BYTE, September 1975. Note that this improved version eliminates any dependence on the clock duty cycle, works directly at higher baud rates, and has no need to worry about phase reversals inside the recorder. The 300 baud version of BIT BOFFER is compatible with the provisional BYTE cassette data interchange standard.

About the Circuit

The block diagram of the BIT BOFFER appears in figure 3 with key wave shapes shown in figure 4. Transmitter and receiver schematics follow in figures 5 and 6. The PC and component layouts are shown in figures 7 and 8.

Our transmitter (figure 5) starts with a 64X frequency reference, or 19,200 Hertz in the case of a 300 baud system. This is divided by four with two type D flip flops set up as binary dividers. The clock is divided by one or by two, as selected by the pair of NOR GATES (IC3a and b), using the TXDATA command out of the serial interface. This selectively divided clock is routed to an eight level sine wave synthesizer consisting of a four stage walking ring counter and a three resistor summing network. The synthesized sine wave at this point has no harmonics till the seventh and ninth, both of which are over 16 decibels down. The synthesizer is followed by a third order Bessell active lowpass filter. This provides a low impedance, one volt peak to peak sine wave at the recorder AUX input.

Note that the UART serial output is inherently synchronized to the UART clock input so that we always switch sine waves just before the leading zero. Feedback from this switching edge is also used to synchronize the divide by four via a pulsed reset generated in the C3-R2 network. This makes sure all signals are properly phased with respect to each other.

On playback (see figure 6), an input RC filter minimizes the hum and bias interference. It then routes the EAR output signal to a high gain limiter that outputs a 5 volt square wave whose zero crossings correspond to the zero crossings received from the recorder. The actual crossings are detected with the Exclusive OR circuit (IC4c and IC4d) that follows. This Exclusive OR circuit outputs one narrow positive going pulse per zero crossing. These pulses are used to continuously discharge a timing capacitor set up as a retriggerable monostable.

Between zero crossings, the capacitor is allowed to charge at a rate set by the BAUD ADJUST control. The CMOS inverter stages that follow (IC3c to IC4b, IC3d to IC4a) act as comparators. Circuit timing is adjusted so that the monostable produces an output at 3/4th the low frequency period, and nothing for the high frequency period. These output pulses are present only during digital zeros and are routed to an output flip flop that regenerates the one and zero data. A lockout on this flip flop, via the RESET input, prevents zeros from being output during leader time, and recorder start and stop intervals. This prevents the UART receiver from reacting to random garbage.

Meanwhile, the recovered clock pulses are added to the new clock pulses that you get from the leading edge of the monostable during zeros-as-data times. These clock and data outputs are routed to the UART receiver serial data (RXDATA) and clock inputs (RXCLK) for recovery.

The .01 μF timing capacitor is set up for 300 baud. This can be doubled for 150 baud or halved for 600 baud operation. Alternately, a larger value pot can be used, perhaps a multiple turn one. Sine wave filter

capacitors in the transmitter portion of the circuit must also be changed for different baud rates.

Checkout and Operating Hints

For your first time preliminary checkout and setup, it's handy to have a scope available; but once you're set up properly with a known good recorder, the circuit essentially takes care of itself with only one non-critical adjustment.

To make an initial test, connect up +5 and ground and connect the reference clock to a source of 19,200 Hertz. This can come from a 555 timer; or, if you're using an existing data rate generator, as a "300 baud, 64X clock" or a "1200 baud, 16X clock." Temporarily break the serial data input to the BIT BOFFER so you can connect it to +5 for a one, to ground for a zero, or to the UART serial output for data.

Set the data input to a one and check for a clean one volt peak to peak 2400 Hertz sine wave at test point A in figure 5. Now switch to a grounded (zero) input and check for an identical sine wave of half frequency.

Next, jumper the BIT BOFFER transmitter output to its own receiver input and check for a clean square wave at test point "C" (figure 6). Now, monitor test point "E". With the data input grounded, adjust the BAUD ADJUST control to get a waveform that is positive between 25 and 30 percent of the time. If you have no scope, use a voltmeter to set the voltage at test point D to a 1.2 to 1.6 volt range on a 5.0 volt supply. Now, switch to an input one, and verify that this voltage and its waveform goes to zero.

The next thing to do is to write a program for your computer which tries sending data through the BIT BOFFER-UART combination, displaying the transmitted data on a Teletype, a TVT, or other output device. Note that if you can't get the BIT BOFFER to talk to itself, there is no way in which it will work when you add a recorder to the middle of the loop. If you have no scope, use the optional but recommended tuning circuit shown in figure 9.

If the BIT BOFFER passes its self transmission test, select a recorder and connect it. If your recorder allows it, monitor the EAR output on a scope while recording. If the sine waves look wrong, try adjusting the input signal level going into the AUX input, or select a different recorder. If the signals going onto the tape look bad, what you get back will be even worse. The BIT BOFFER should properly "echo" the message during recording, provided your recorder has output to the EAR jack for record monitoring.

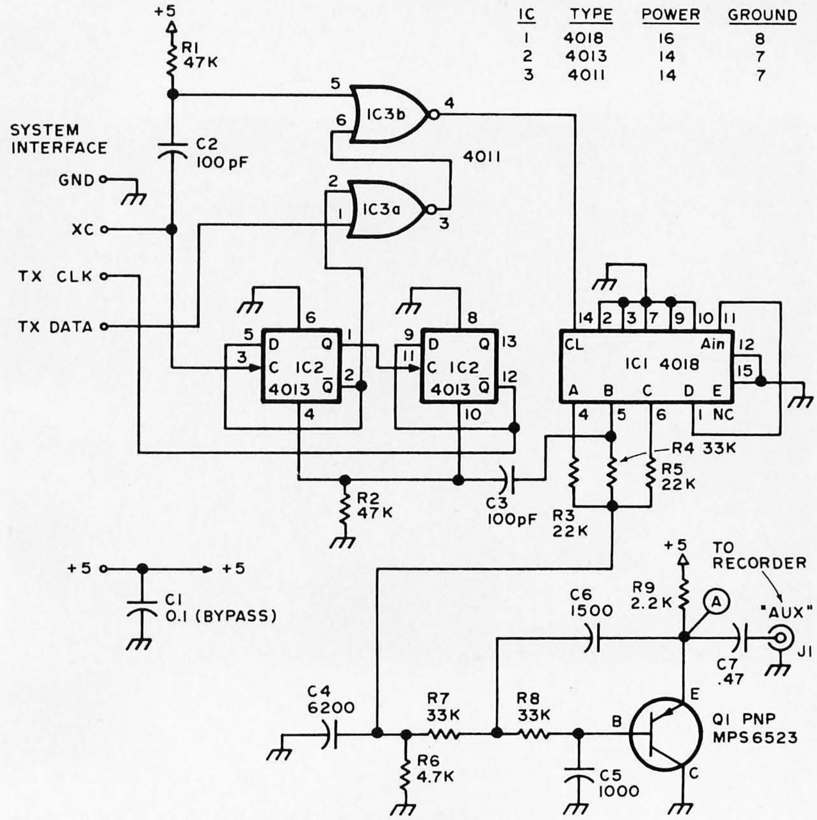


Figure 5: Schematic diagram of the BIT BOFFER transmitter section.

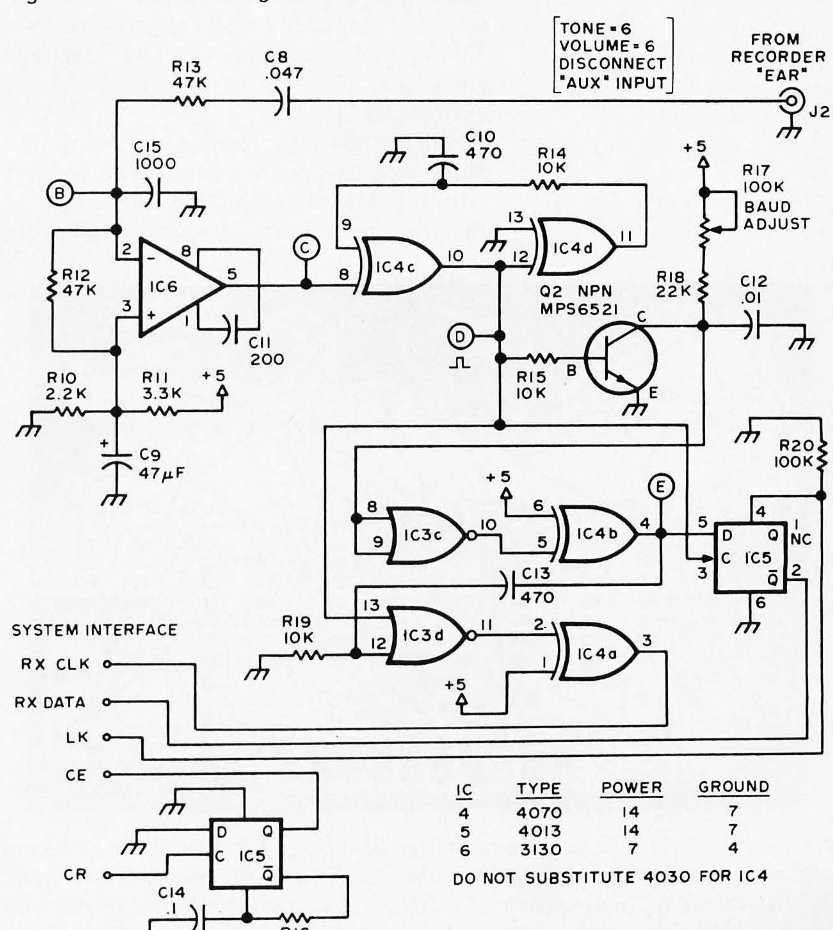


Figure 6: Schematic diagram of the BIT BOFFER receiver section.

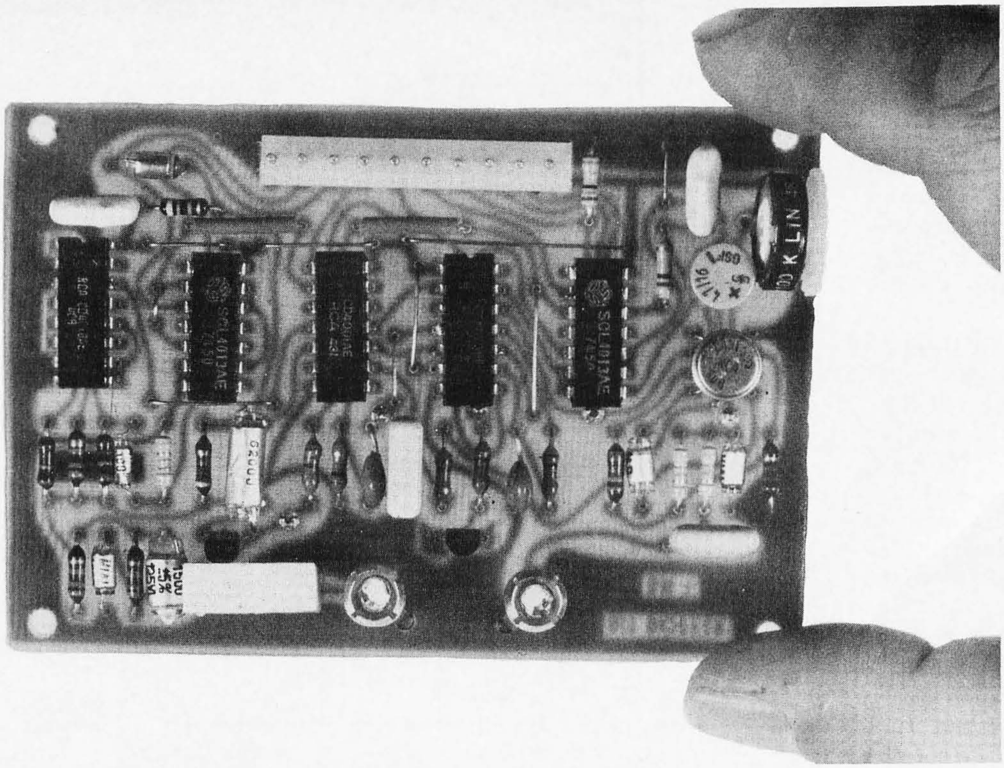


Photo 1: The assembled BIT BOFFER built from the board layout of figure 7. The parts can be identified using the placement diagram of figure 8.

Now we're ready to record some data. Use a high quality, audio cassette tape, and record a character sequence generated by a TVT or a microprocessor program. [See Jack Hemenway's article in this issue for an example. *Ed.*]

During playback, monitor test point "B". Some recorders get unhappy if you connect both the AUX input and the EAR output at the same time during playback, particularly on line operation. If yours does, *unplug the AUX input during playback*. Check for reasonable appearing sine waves at test point "B". Try various settings of the volume control and tone controls. Usually settings of "5" for volume and "7" for tone are optimum.

Now, set up an oscilloscope and look at the *eye diagram* as shown in figure 10. You should get this same pattern at test point "C", with only a small amount of jitter. Set your BAUD ADJUST control to sample in the center of the right eye. If you can't get the jitter down far enough that this sampling is perfectly clean, stop and find out why. Use the optional tuning circuit shown in figure 9 if you have no oscilloscope.

If all this works, you should now be able to reliably record and playback data of your choice. Above all, get to know your recorder and know what its limits are in the way of input level, tone, jitter, and volume settings. Note particularly that any second harmonic distortion of the recorder must be held down to something reasonable or alternate zero crossings will jitter. *Your eye diagram tells all about the quality, jitter, and error rate of your system.* Learn to use it. (By the way, don't make the mistake of using a dual channel scope on alternate vertical inputs, as this can mask any second harmonic jitter. Use a single trace or else chop the display.) Note that you can get continuous data for debugging by connecting the receiver RDY output to the transmitter KP input. You

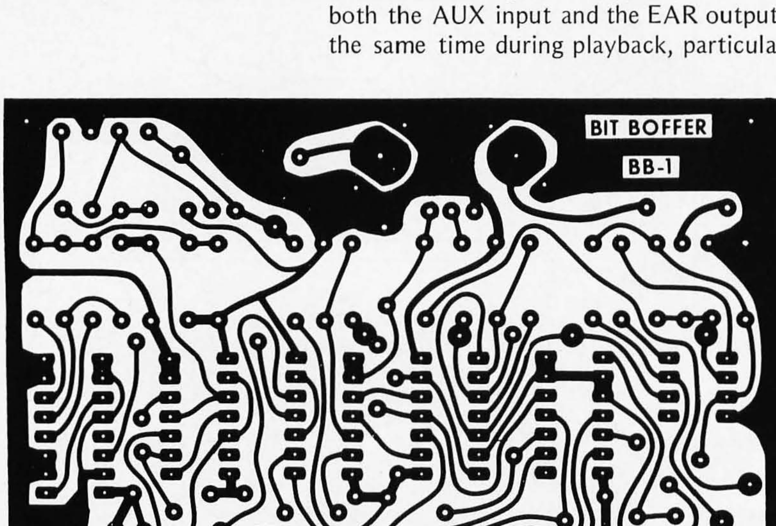


Figure 7: Printed circuit layout of the BIT BOFFER modulator. A board with this pattern is available from Southwest Technical Products Corporation, 219 W Rhapsody, San Antonio TX 78216. Also available is a kit of parts for the BIT BOFFER, a kit which contains a UART interface to mate with the SWTPC 6800 system, and a kit of parts for the tuning indicator.

Table 3: Error rate hardening techniques for premium systems.

***CLOCK PLL**
A phaselock loop on clock output can fill in occasional missed clock pulses. Present system only allows six misses per word.

***DATA INTEGRATION**
Analog integration or speed-independent majority logic voting takes average of several sequential half-cycles. Present system depends only on UART sampled mid-bit word.

***ERROR FLAGS**
Parity, overrun, and framing error flags already in UART circuit can be used to stop loading if error occurs.

***FULL CR DECODE**
Some TVTs interpret any machine command as a carriage return. Limiting decoding to proper command minimizes severity of display errors.

***LOCKOUT**
Existing lockout circuit can be used to disable reception during tape start, leader, and stop times.

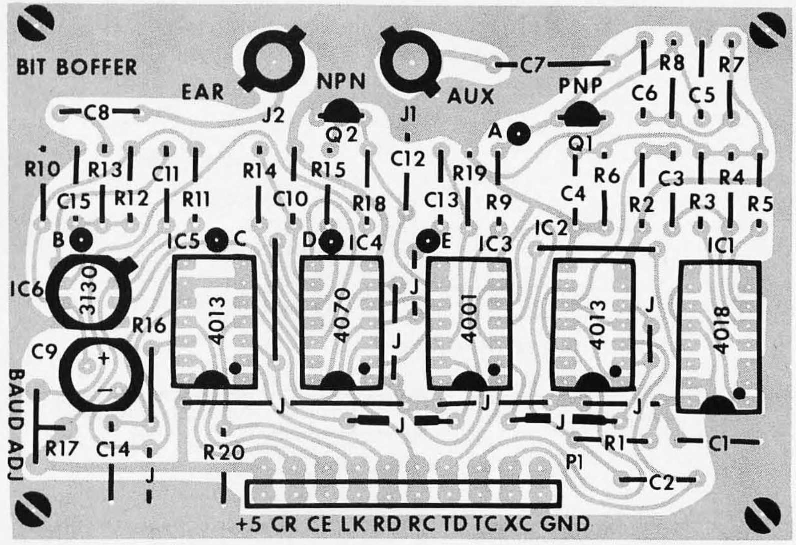


Figure 8: Parts placement overlay for the BIT BOFFER circuit board.

may have to *momentarily* short this line to ground to start the action.

Once you're confident of your BIT BOFFER, you're ready to exchange data. If you get an outside cassette, it should work without adjustment, if it was recorded properly. If you get errors, simply use your tuning indicator and rotate the BAUD ADJUST control to note the two limits where serious errors start, and then set the pot halfway between these limits. Better yet, use your scope and the eye diagram to properly set this control. In actual day to day use, this pot is quite uncritical. Don't forget to change capacitors or use wider range control if you make wide changes in the data rate.

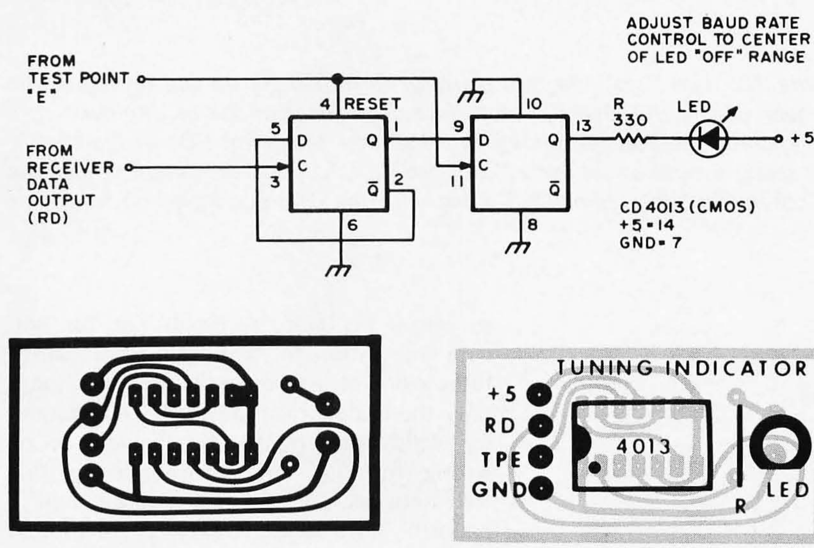
Some Modifications

"Make it cheaper" and "make it better" are usually the name of the game in any new electronic system. What can we do to further improve the BIT BOFFER?

The error rate seems acceptable for most hobbyist interchange uses as is, but there are many things you can do to "harden" the error rate for a fully professional and reliable system.

Some hardening techniques are listed in table 3. Careful adjustment of the baud rate control, using the tuning indicator on a scope eye diagram, is a first and foremost line of defense against errors. You can add an output phaselock divider loop to reconstruct and more evenly space the UART clock pulses. This fills in for an occasional miss. The present system allows up to six or

Figure 9: Tuning circuit board, overlay and schematic.



A TUNING INDICATOR

If you don't have an oscilloscope for your initial testing or don't want to tie one up for eye diagram testing of incoming tapes, you can use this simple tuning indicator instead. When the light emitting diode is out, your data rate control is properly set. Adjust the data rate control to the CENTER of the off portion of the display. The light emitting diode should stay off while good data is being received.

If your data rate control is set slightly too high or too low, you'll get one too many or one too few clock pulses for each string of zeros. The tuning indicator works by checking for an odd or even number of monostable pulses during zero times. It then stores this odd-even information between zero strings. Very nicely, the tuning indicator will often indicate trouble BEFORE it causes errors, thanks to the UART being able to withstand a few extra or a few missing clock pulses per word without error.

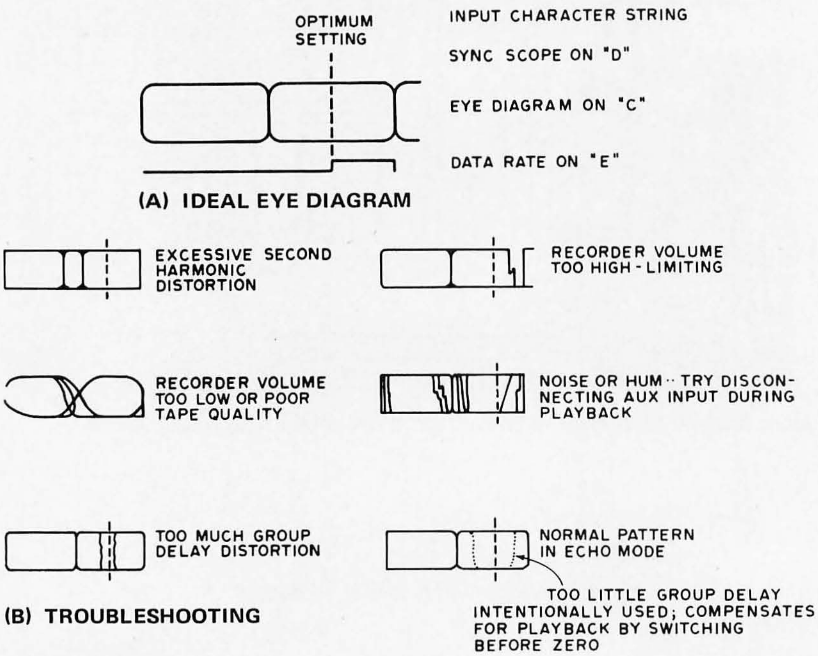


Figure 10: The "eye" diagram is useful for initial set up and testing of the receiver of the BIT BOFFER interface. This diagram can be displayed on a two-channel oscilloscope set up to trigger on test point "D" in the circuit. The scope should be set for a "chopped" display, and the sampling edge can be observed at test point "E" along with the data eye found on test point "C".

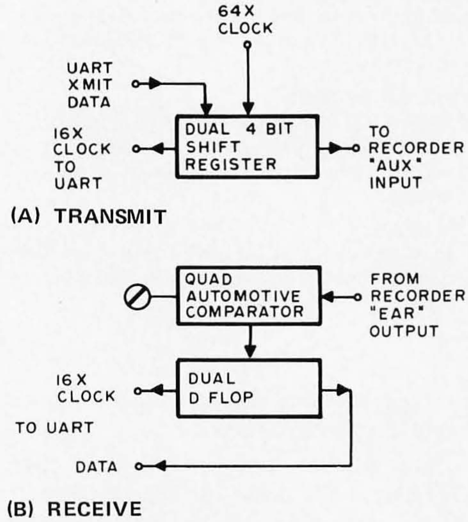


Figure 11: A possible future development of the system is a simplification of the circuit using only three integrated circuits.

so misses per word received. You can add data integration to "vote" on what constitutes a one or a zero. In the present system, only the middle bit is all the UART samples for valid data. If this bit happens to be wrong, you lose. Voting can be done with a RC network, or a shift register and a majority logic gate. The analog method is cheaper, but the shift register scheme is data rate independent.

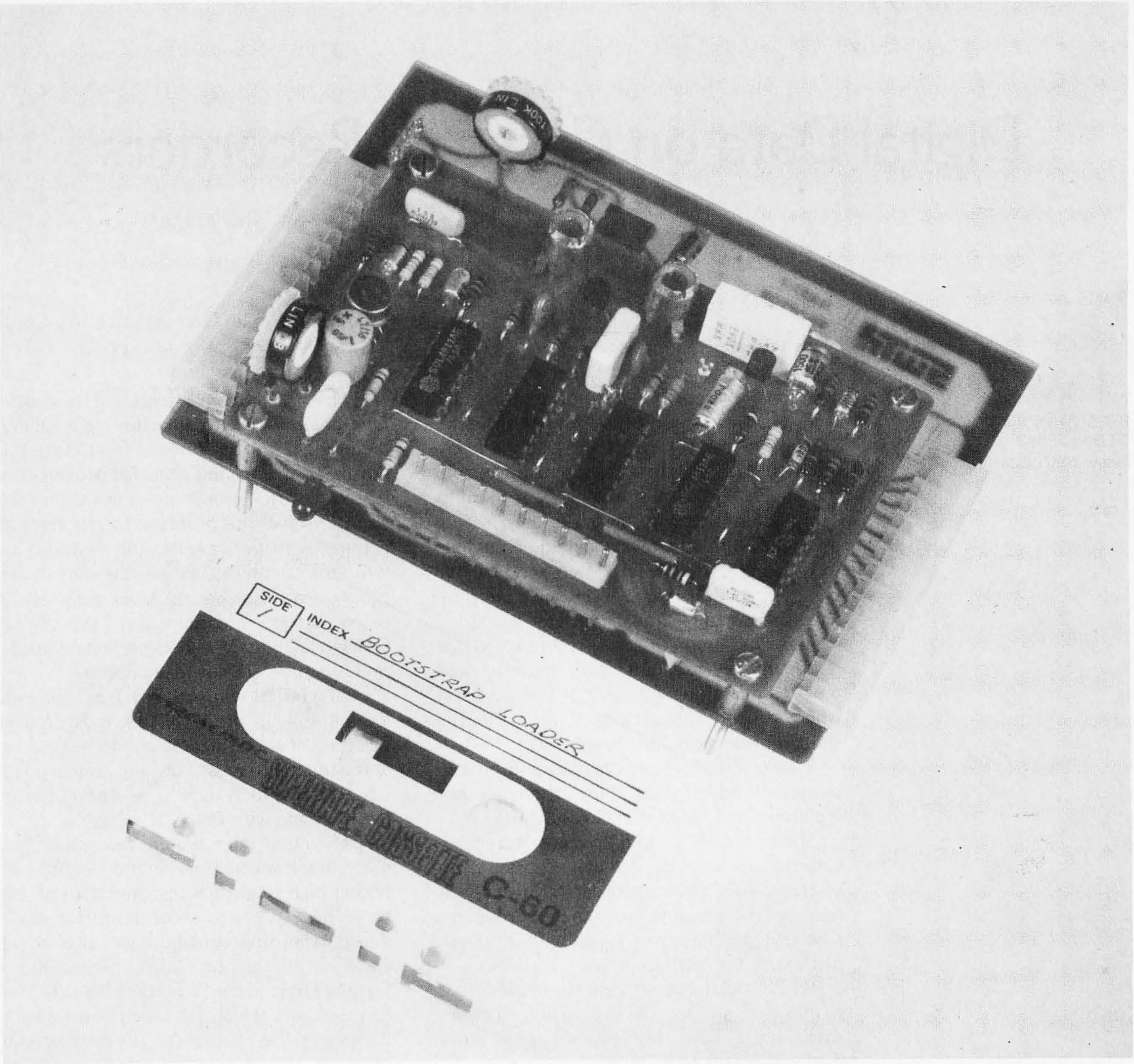
You can use your lockout to prevent data entry during recorder leader, start, and stop times. You can use your existing error flags on the UART output to automatically stop TVT data entry or playback if a parity, framing, or overrun error happens. As noted earlier, these flags can be used with your computer software.

If you have a TVT that uses any control signal as a carriage return, add a decoder to detect only the valid CR signal as a return and ignore all other machine commands. This eliminates errors that tear up the whole display. For instance, a single long tape dropout can be read as an ASCII null command. It gets ignored with a full decoding but defaults to a carriage return if you don't do your decoding properly.

If you're running your BIT BOFFER above 600 baud, you can try making the high frequency sine waves bigger by pre-emphasizing them. This is simply done using a CMOS 4066 analog switch to change the load resistor on the sine wave synthesizer. Pre-emphasis can eliminate much of the recorder's differential gain to higher frequency signals. Since most automatic level controls are a fast attack, slow release type of device, they won't excessively interfere with this particular type of amplitude modulation, so long as the normal or mark state corresponds to the higher frequency and the higher amplitude.

You can probably come up with some other schemes for further error rate hardening on your own.

Can we make it cheaper and simpler? Figure 11 shows a three IC system that's still under evaluation. The receiver uses a quad automotive comparator to perform zero crossing detection (1 and 2), reset the timing capacitor (3), and sense the capacitor voltage



(4). This is routed to a dual D flop that detects ones and zeros with one side and combines the clocks with the other. A dual four bit shift register is one possible route to a simpler transmitter. The actual circuits are still under test.

Let us know your experiences with the BIT BOFFER system so that further improvements can be made, particularly to handle your particular uses for this simple, speed tolerant, software independent cassette interchange system.■

Photo 2: This is how the BIT BOFFER mates with a Southwest Technical Products Corporation UART adapter board in a "piggy back" fashion. See the caption of figure 7 for information on where to get the interface parts.

Digital Data on Cassette Recorders

(The Demise of an Overworked Carry-Corder)

Harold A Mauch
Pronetics Corp
PO Box 28582
Dallas TX 75228

Nearly everyone has a portable cassette recorder. If you don't have one, chances are your kid does ("Hey, Mom, Dad stole my tape recorder!"). These recorders range from the under $20 "bare bones" variety to multi centibuck units with nearly every feature imaginable. Fortunately it should be possible to use nearly *any* cassette recorder available if it is clean and in good working condition. Pawnshops and similar outlets are good sources of used cassette recorders. Used recorders are often quite dirty and may need repair. Take along a couple of test cassettes when you go shopping and check out the units' operation before buying.

Watch out for bent capstans and broken cassette holders since these often are not repairable and indicate excessive abuse.

Some dictating and "pocket secretary" cassette recorders do not use a capstan drive system. While these recorders are usable, it may not be possible to exchange programs recorded on these machines with a friend. Stick with the capstan driven recorders.

While nearly any cassette recorder is usable for storage of digital information, some units have features which improve performance or convenience.

Tops on the convenience list is a *digital tape counter*. Next to destroying a valuable recording, nothing is more frustrating than not being able to find a desired program on a cassette with several programs. The tape counter solves this problem. Merely reset the counter with the cassette fully rewound and write the counter reading of the start of each program on the cassette label. Some of the newer cassette recorders also have cue and review capability. While occassionally useful, these features are not really necessary.

A recorder with an AC *bias and erase oscillator* will produce the most reliable performance and highest quality recordings. Unfortunately most of the under $100 cassette recorders now available erase and bias the tape with DC.

DC erased and biased recordings have more low frequency noise and residuals and poorer high frequency response than AC bias recordings. Cassette recorders designed for music recording usually have circuitry to erase and bias the tape with a 50 to 100 kHz signal. These same recorders usually have drive motors which are speed controlled by the power line frequency. The result is more precisely driven and recorded tape. Since

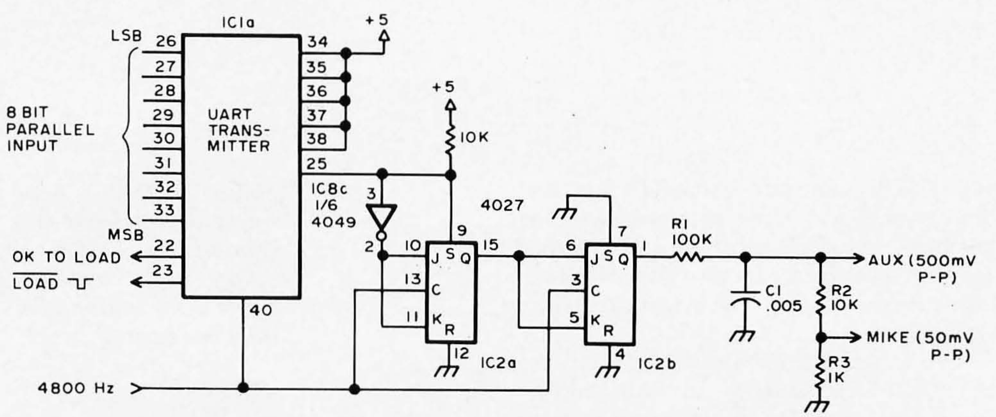


Figure 1: Cassette digital modulator. This circuit converts 8 bit parallel data from a computer into a series of 2400 Hz and 1200 Hz tones using a UART. Filtering provided by C1 and R1 is used to turn the square wave outputs of IC2b into a closer approximation of a sine wave (see figure 2).

these are normally stereo recorders, be sure to bulk erase the cassette first to remove the residual signals between the stereo tracks. If you apply the signal to be recorded to both channels, the resulting recording will be usable on any of the portable cassette players.

The cassette tape unit you select must have an auxiliary (AUX) or microphone input and a line or earplug output. This is the only reasonable way to connect the cassette tape unit to the necessary modulator/demodulator circuitry. Acoustic coupling through the microphone and speaker is totally unsatisfactory.

Pause controls are nice but not necessary.

Use the cassette tape unit available to you, but remember you only get what you pay for and these days even that costs more.

Choice of Cassette and Tape

The choice of cassette cartridge and tape has more effect on performance than *all* other factors combined. This is no place to save a penny or even a buck. Get the very best tape you can buy. Do not even consider anything less than the super tapes. If your recorder can record the chromium dioxide tapes, use them. Anything less than the best will result in much frustration. Avoid using the C90 and C120 cassettes. The tape is too thin and fragile. C60 and shorter tapes are much more rugged.

If a cassette is not in use it should be stored in its container in a dust free location. Keep the cassette tape unit spotlessly clean and do not smoke in the room in which the cassette equipment is used or stored.

It is impossible to adequately stress the importance of buying the very best quality tape and then keeping it and the tape unit clean. Tape quality and cleanliness is much more important in digital applications than in the more conventional speech or music applications.

Getting the Digital Information onto the Cassette

There are many ways to record digital information on audio cassette tapes. Many of these techniques work quite well as long as the data is played back on the same machine as was used to make the initial recording. Rather than debate the merits and deficiencies of the various techniques, the author has chosen to support the proposal suggested for evaluation by the BYTE sponsored symposium on audio digital cassette recording. I feel the proposal adequately accommodates the limitations imposed by conventionally available audio cassette tape units.

Digital information from your computer is generally available as 8 bits parallel from an IO port or data bus. The recording on tape must be serial with start and stop delimiting bits. The transmitter portion of the UART is ideal for converting the parallel data to this serial format. Figure 1 is a circuit implementing such a converter or modulator.

The serial output of the UART is said to be NRZ (non return to zero). It means that a logic one bit is a high level and a logic zero bit is a low level. A logic one causes the modulator to generate a 2400 hertz output signal and a logic zero generates a 1200 hertz signal. Normal output from the modulator is a string of square waves. The sharp edges of the square wave signal do not usually record well on recorders with DC recording bias. The designers of such recorders "roll off" the amplifier low frequency response and boost high frequency response in an attempt to diminish the drawbacks of DC biased recording. This causes a square wave to be abnormally "peaked" on the rising and falling edges and the flat portions to be "tilted." Refer to figure 2.

Such signals are more likely to cause errors during playback. Ideally the modu-

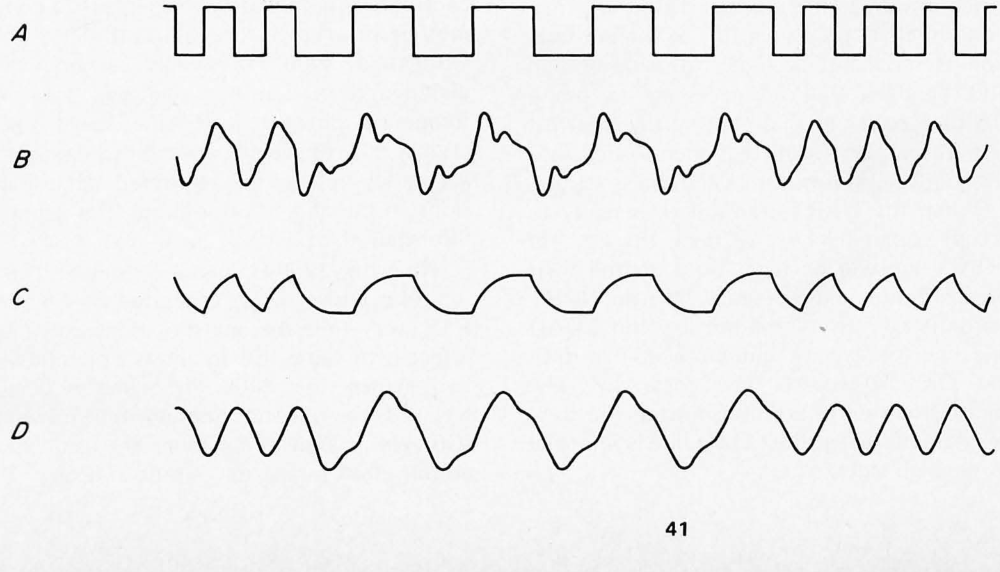


Figure 2: If a square wave signal such as waveform A is recorded on a low cost cassette recorder, the playback response may look like waveform B, which is very difficult to demodulate. If the square wave is filtered with a low pass filter before recording (waveform C), the playback response will look like waveform D, which is a usable signal.

lating signals should be sine waves but generating and switching sine wave signals digitally is somewhat complicated. "Rounding the square wave corners" with a low pass filter (R_1 and C_1 in figure 1) is not totally effective but does provide a usable waveform.

The AUX output is a 500 mV peak to peak signal. This signal level will overdrive a microphone input and should only be connected to the recorder auxiliary input (50 kOhm or greater input impedance). The MIKE output is 50 mV peak to peak and will drive most cassette microphone inputs.

The 4800 Hz signal should be as precise as possible and capable of driving 2 TTL loads. Ideally it should be obtained from a crystal oscillator and divider string or a phase locked loop (PLL) locked to the power line frequency. If such stable sources are not available the circuit shown in figure 3 is satisfactory but it must be accurately adjusted with a frequency counter.

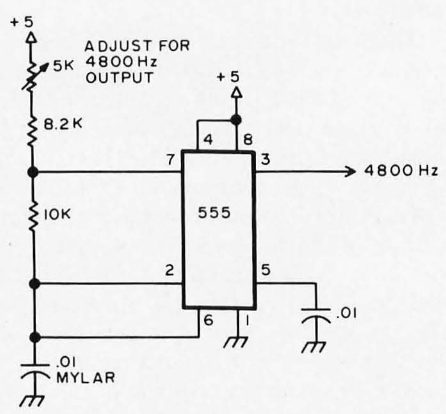


Figure 3: Circuit of a 4800 Hz oscillator. This oscillator, using the 555 precision timer circuit, can be used if a crystal controlled or line frequency derived timing source is not available.

If the available digital information to be recorded is already in serial form with the necessary start and stop bits (2 stop bits are required) and is being sent at 300 baud, the UART transmitter is not necessary. However, the 4800 Hz clocking signal should be synchronous with the serial digital information (16 clock pulses per bit). If the information is serial but at some rate slower than 300 baud, it will be necessary to use a UART receiver to first convert the information to parallel form. It is then loaded into the UART transmitter as described earlier.

When the UART transmitter is ready to accept a parallel byte of data, the OK TO LOAD line will be high. Data on the eight parallel input lines is loaded into the UART transmitter buffers by pulsing the LOAD line low for at least 1 microsecond or until the OK TO LOAD line goes low. The transmitter will start transmitting the byte or character when the LOAD line is returned to the high state.

If the UART is not transmitting any data, its serial output line is high, causing the modulator to generate the 2400 Hz signal.

Playback of the Recorded Data

Since the signal recorded on tape is basically a standard FSK (frequency shift keyed) signal, it is possible to recover the digital signal with a phase locked loop (PLL) or FM discriminator. In fact, users of the Suding cassette system (wide shift audio FSK) should be able to recover the NRZ data signal by readjusting their demodulators. However, data recovery by these means is not as precise nor as insensitive to tape speed variations as digital recovery techniques which extract speed insensitive timing pulses from the recorded signal and use these pulses to retime the NRZ data.

Figure 4 is a complete schematic of the playback recovery circuit or demodulator.

The cassette earplug output signal is conditioned by the operational amplifier Schmidt trigger IC3. IC4 is a retriggerable one shot with a period of 555 microseconds. As long as the 2400 Hz signal is being received, the one shot is constantly retriggered and does not time out. This causes flip flop IC5a to remain at the high state interpreting the data as a logic one. When the 1200 Hz signal is received, its period is long enough to allow the one shot to time out. Flip flop IC5a is immediately reset. It stays at the low state as long as the 1200 Hz signal is being received, because the one shot is timed out whenever the next triggering edge occurs. When the 2400 Hz signal returns, the one shot output stays high, thereby permitting the flip flop IC5a output to switch to its high state. The output of flip flop IC5a is the recovered NRZ serial data.

Under ideal circumstances, the recovered data would be sufficiently stable to drive a 300 baud teleprinter or TV typewriter directly. However, if the tape speed varies in excess of approximately ±6 percent (a common occurrence), errors will result. Since the 1200 and 2400 Hz signals carrying the digital information on tape will vary in frequency directly with tape speed variations, it is possible to use these signals to accurately retime the recovered data. Flip flops IC6a and IC6b extract this timing information.

When the 1200 Hz signal is received, IC6a is preset with a pulse generated by C8 and R15 every time the one shot times out. The effect is to cause IC6 to act as a division by two. When the 2400 Hz signal is being received, the one shot does not time out and IC6 acts as a divide by four. The result is a double clock rate at the output of IC6b.

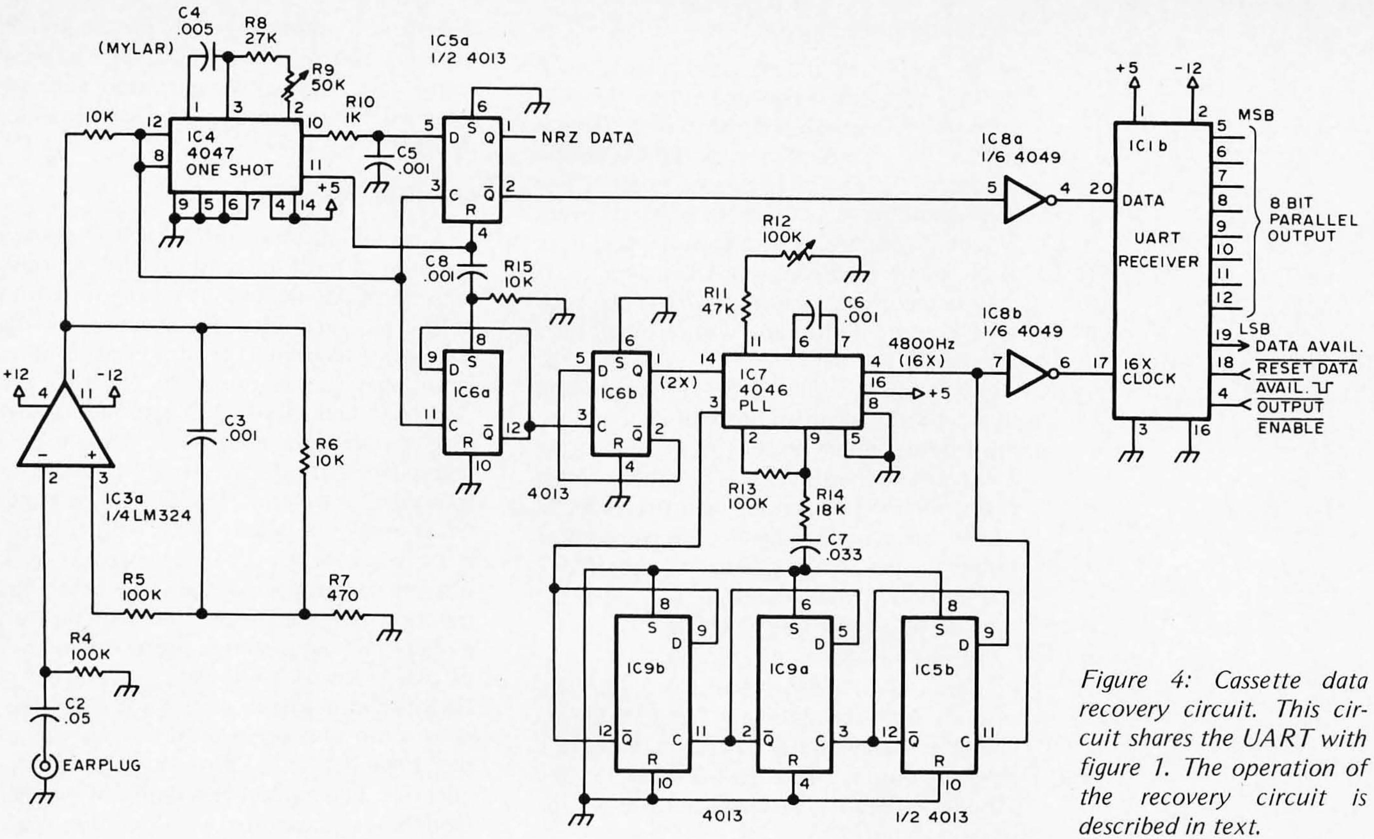


Figure 4: Cassette data recovery circuit. This circuit shares the UART with figure 1. The operation of the recovery circuit is described in text.

Instead of clocking the data into a shift register, it may be more desirable to use the receiver portion of a UART, since the UART receiver has built in circuitry to identify the beginning and end of each byte or character automatically. Furthermore, the UART parallel data outputs are 3-state, which permits convenient direct connection to most IO ports or data buses. (For a more detailed discussion of the UART, you may wish to read "Serial Interface" by Don Lancaster in BYTE, September 1975).

However, the UART requires a clock at 16 times the data rate. This problem is solved by phase locking an oscillator at 4800 Hz to 600 Hz (2X) ouput of IC6b. The phase locked loop (PLL) oscillator is adjusted for 4800 Hz in the absence of any input signal. IC5b and IC9 divide the PLL oscillator output by eight and drive one of the PLL phase detector inputs. The other phase detector input is driven by the 2400 Hz clock output of IC6b.

When the UART receiver recognizes that it has received a complete character, it raises its DATA AVAILABLE output line to logic one (high level). Since the UART outputs are 3-state, it is necessary to drive the RECEIVED DATA ENABLE input to logic zero (low level) to read the parallel output data. After the parallel data has been read, it is necessary to pulse the RESET DATA AVAILABLE line to prepare the UART to output the next byte or character. The pulse must remain at logic zero for a minimum of one microsecond or until DATA AVAILABLE drops to logic zero.

Circuit Adjustments

As already stated, the 4800 Hz signal used to drive the UART transmitter and modulate the tape recorder should be obtained from a very stable and accurate source for best results. No other adjustments are necessary on the recorder modulation circuits.

The data recovery one shot and the phase locked loop oscillator in the playback data recovery circuits must be accurately adjusted for best results. The most critical adjustment is the period of the data recovery one shot. An easy way to adjust the period is to connect a well calibrated audio oscillator to the earplug input of the data recovery circuit and a high impedance voltmeter to the NRZ data output (IC5a pin 1). Set the audio oscillator for 1800 Hz and the output level for 1.5 to 3.5 volts RMS. Adjust R9 until the voltmeter reading just changes (use the 5 to 15 volt scales). Get the adjustment as close to the point of change as possible.

The PLL oscillator is adjusted for 4800 Hz (R12) with no connection to the earplug input. If a frequency counter is not available, compare the PLL oscillator output (IC7 pin 4) to the 4800 Hz signal used to drive the UART transmitter.

Operating Procedure

The playback data recovery circuit will operate best with an earplug output signal of between 4 to 10 volts peak to peak. This is within the range of most portable cassette recorders. It may be necessary to put a low gain amplifier ahead of the data recovery circuit if you are using a cassette tape deck not capable of driving a speaker directly. It may be necessary to turn down the playback tone control if the tape was recorded on a DC biased recorder.

To comply with the BYTE Symposium Standard, the recorded block of data on tape must have a minimum of five seconds of the 2400 Hz tone before data is recorded. This is easily obtained by permitting the recorder to run in the record mode for five seconds or longer before sending data to the UART transmitter. When the UART is idle the modulator is generating 2400 Hz.

During playback it is recommended that you wait until the playback is one or two seconds into the 2400 Hz "leader" before allowing the computer to accept the UART receiver output. This is to avoid reading "trash" caused by turning the cassette tape unit on and off.

It is possible to turn the cassette tape unit on and off with a relay under computer program control using the cassette tape unit remote control input. However, the cassette will record and playback "trash" during the startup and stop intervals which may take as long as 3 to 5 seconds. The 2400 Hz signal recorded on tape before each block of data gives the computer a "trash free" interval in which to prepare itself for the data to follow.

Circuit Design Considerations

It will be some time before enough information has been learned about the use of audio cassette recorders for storage of digital information to permit truly optimum designs of the necessary modulator/demodulator circuits. Therefore the author would like to present his design considerations to provide other experimenters and designers a starting point for additional experimentation and optimization. The comments are somewhat technical and are intended for the advanced experimenter or designer.

Modulator Waveform

The nonlinearity and skewed frequency response of most low cost cassette recorders impose serious limitations on the waveform of the recorded signal. In severe cases, the waveform recovered from a square wave input may be so seriously "tilted" and "peaked" and filled with overshoots that data recovery is impossible. Obviously a better modulating signal would be a sine or triangular waveform. On the other hand, "doctoring" the square wave with filters is attractive from an economic viewpoint. Such filtering can only be carried so far before the resulting differential amplitude of the two modulating frequencies produces "pumping" of the recorder automatic level control circuits and begins to diminish the signal-to-noise ratio and signal drop out margins of the higher of the two modulating frequencies. Economical generation of a better modulating waveform will go a long way toward improving data recovery reliability with simple recorders.

Modulator Signal Level

The signal level applied to the recorder appears to be relatively uncritical. However, I feel the level should be standardized; but I am not prepared to recommend a preferred level at the present time.

Demodulator Signal Conditioning

Many experimenters have used simple zero crossing comparators to condition the playback signal. While these circuits have tremendous immunity to signal drop out, they are quite sensitive to "drop in noise" and tend to "chatter" at low signal levels or in the absence of an input signal. I prefer a circuit with sufficient hysteresis to provide some margin against the drop in noise and residuals and to prevent chatter. The ideal

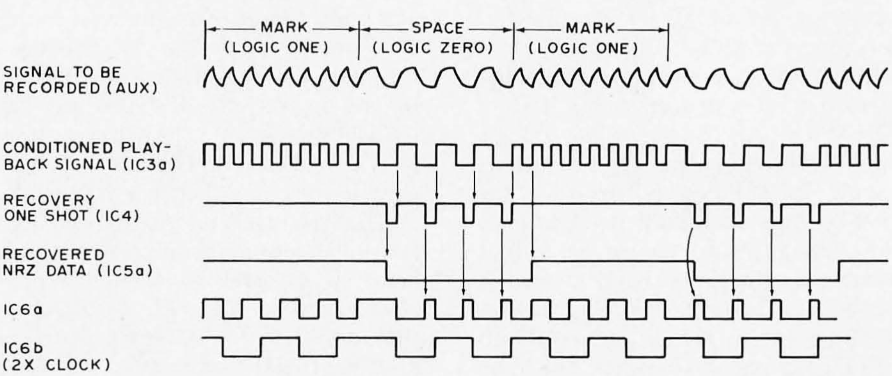


Figure 5: Cassette modulator demodulator wave forms. The signal presented to the tape recorder is a filtered square wave, shown at the top. The timing of data recovery is shown relative to the conditioned playback signal in the remaining five traces.

trip points for such a circuit is probably in the range of 20 to 30 percent of peak signal. The trip points of the circuit described in this article are approximately ±0.5 volt. Best performance will then be obtained from 3.5 to 5.0 volt peak to peak input signals.

Demodulator One Shot

If the one shot is properly adjusted, the data is recoverable with tape speed variation in excess of ±30 percent from nominal speed. I have found the speed distribution of the portable cassette tape units to be skewed roughly 5 percent negative. If a tape is played on the same unit as was used to make the recording the problem is negligible. If, however, the tape was prepared on precision tape recording equipment (such as may be used for mass production of cassettes for widespread distribution), then played on a consumer quality tape player, the tolerance of the recovery circuit to a decrease in tape speed will be diminished. This may provide some argument for increasing the period of the one shot 5 percent.

A characteristic of the data recovery circuit used is that it causes an approximately 6 percent marking bias in the recovered waveform. This is not too important if the data is recovered by a shift register or clocked into a UART receiver. A purist approach would delay the space to mark transition 6 percent of the nominal bit cell duration.

Some experimenters filter the recovered data waveform to provide an additional immunity to error. I have not found it to be necessary and have found it creates more problems than it solves.

Demodulator Phase Locked Oscillator

The PLO is only necessary because the UART requires a clock at 16 times the data rate. The phase detector output is filtered with a lag-lead network. The filter was designed to permit capture of signals ±15 percent from nominal speed with a 0.707 damping factor. Consequently, the oscillator will remain locked during ±15 percent step changes of the input signal frequency. Once locked, the oscillator will track the input signal over a ±70 percent range. The sum frequency component of the phase detector output does modulate the oscillator slightly but was not considered to be a problem. This modulation can be diminished by increasing the loop filtering; however, this reduces the capture range which is undesirable.

Conclusion

The use of hardware to modulate and demodulate the cassette tape simplifies the programming problems associated with using the cassette for program loading and storage. In some circumstances it may be possible to connect the cassette hardware interface directly to your panel switches and display drivers and "let it rip." Other systems may require peripheral interface adapters or other similar circuitry to get the data onto and off the computer data bus.

The cassette interface described in this article is manufactured by Pronetics Corporation. It is available fully assembled and tested on a 4.5 x 6.5 inch circuit card with connections through a standard dual 22 pin gold plated card edge connector. Price, availability, and other information may be obtained by writing: Pronetics Corporation, PO Box 28582, Dallas TX 75228. ■

Microprocessor Update:

Robert Baker
34 White Pine Dr
Littleton MA 01460

The CP1600 is a complete, 16 bit, single chip, MOS-LSI microprocessor available directly from General Instrument for $99 in single quantity. It utilizes third generation minicomputer architecture as shown in the block diagram in figure 1. Figure 2 shows the actual pin assignments of the 40 pin dual-in-line package. Standard operating voltages required are: +12 V, +5 V, and −3 V DC. The simple bus structure is TTL compatible and allows direct memory access (DMA) capabilities for high speed data transfers.

Microprocessor

There are eight high speed, general purpose 16 bit registers available with R6 reserved as the stack pointer (SP) and R7 reserved as the program counter (PC). The stack pointer (R6) indirectly addresses a dynamic last in, first out storage area called a stack. It is used to hold interrupt and nested subroutine return addresses as well as general data. This provides unlimited stack

PIN ASSIGNMENT
40 PIN DUAL IN-LINE PACKAGE

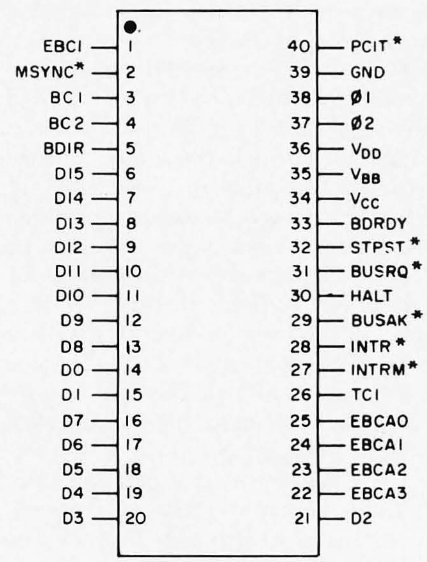


Figure 2: The pin connections of the CP1600 microcomputer. This 16 bit general purpose computer is contained in a 40 pin package with multiplexing of address and data information.

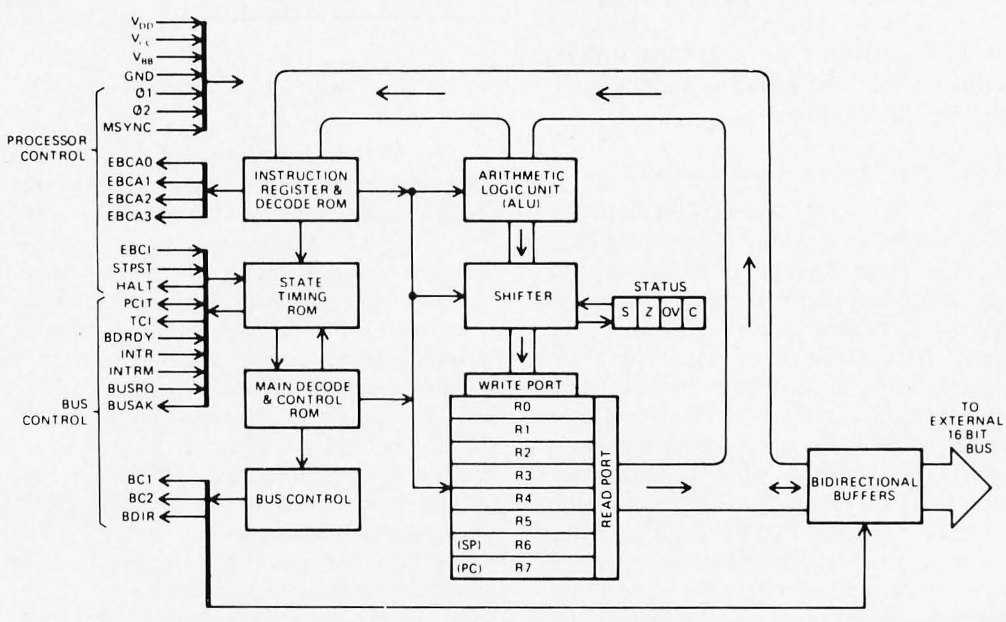


Figure 1: Internal block diagram of the CP1600. This machine uses a general register architecture with two of the eight available registers dedicated to the stack pointer and program counter functions.

General Instrument CP1600

depth and self identifying nested interrupt or subroutine capabilities using external RAM memory.

A 5 MHz, 2 phase clock provides a microcycle time of only 400 nanoseconds. Thus, register to register operations take only 3.6 microseconds while memory to register and input/output operations require only 4.8 microseconds. Full instruction execution times range from 1.6 to 4.8 microseconds.

Four addressing modes (immediate, direct, indirect, and relative) with a 16 bit word length allow direct addressing of 64 K bytes or 32 K words of memory or peripheral devices. Since a single address bus structure is used, both memory and peripheral devices reside in the same address space. Only the user defined system address allocation differentiates memory from IO devices.

No special input/output instructions are required since IO data is manipulated just like memory data using any of the 87 available, general purpose instructions shown in Table 1.

The basic instruction word format consists of 10 bits located in the lower order bit positions of a 16 bit processor word. The high order six bits of every 16 bit word supplied to the processor as an *instruction word* are ignored by the internal microcontrol logic. Thus, a single 10 bit wide ROM instead of dual 8 bit wide ROMs may be used where ultimate ROM bit efficiency is desired.

The branch on external condition (BEXT) instruction allows up to 16 external digital signals to be sampled by the program with a program branch executed if the test is true.

Continued on page 50

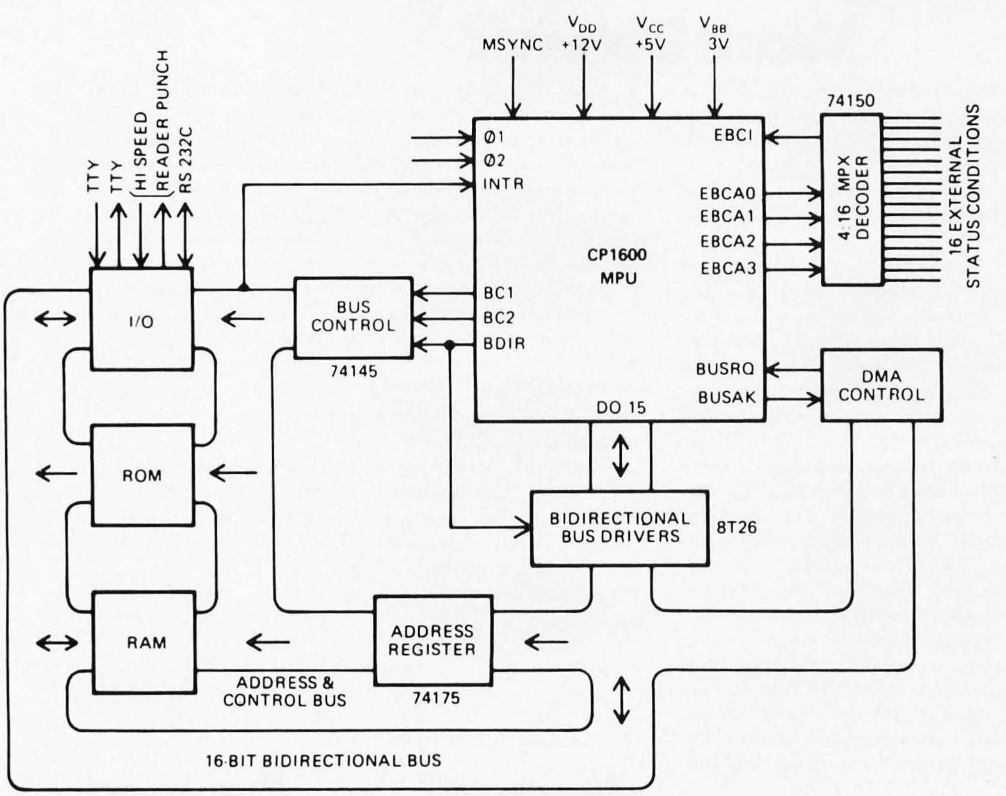
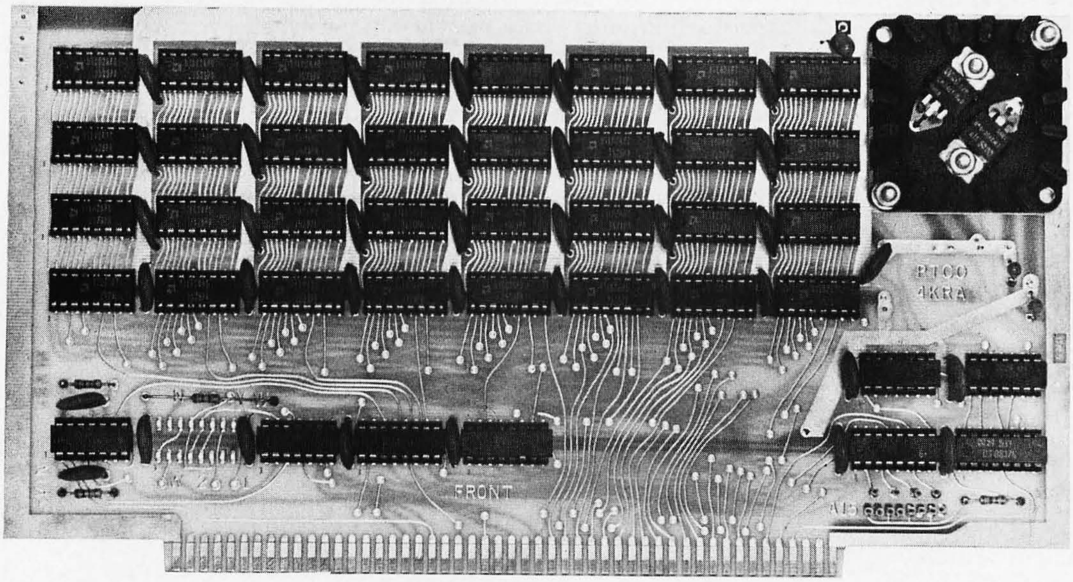


Figure 3: External block diagram of a CP1600 system. This illustrates how the CP1600 is typically combined with other components to produce a computer system. An external address register is required to demultiplex the address information, which is shared on the same 16 bit bus with ordinary data transfers.

4KRA LOW POWER MEMORIES AT A NEW LOW PRICE!



Want Details?

The Processor Technology 4KRA-4 is the fastest, most reliable and yet, least expensive read/write memory module on the market today. The 4KRA-4 is the only memory available of any kind that can be powered by two "D" size batteries for up to ten (10) hours. (Ask for our TB-101 technical bulletin on back-up battery operation of the 4KRA-4).

Don't be mislead by "undirected" statements on the subject of memory power consumption. All RAM's used in the 4KRA-4 consume typically 1/3 the power of 8101 or 2102 type memories. Under absolute worst case conditions our RAM's require only 30% more power than the "typical" consumption of any 4K dynamic memory. Remember that dynamic memories use three power supplies, but static memories only one.

It's time to clear the air of any confusion anyone might have about memory speed. In any 8080 system, all memories with access times between 50 and 520 nanoseconds are the same speed! Access time alone is not a valid indicator of speed unless it is greater than 550 nanoseconds, thereby requiring slow-down "wait" states. However, two other factors affect overall system speed.

1) Dynamic memories must refresh themselves periodically, slowing down the micro processor. In a well designed dynamic memory system refresh slows the processor by a few percent. Static memories do not require any refreshing. When our 4KRA-4 memories are used in the Altair 8800 the "wait" light goes out, indicating maximum speed operation.

2) Long cycle times can slow the system down during critical operations such as Direct Memory Access. DMA is used by most disk memories and by such devices as a soon to be announced color graphics generator. Most dynamic memories now on the market have a 1500 nanosecond cycle time, about three times that of our 4KRA-4. This longer cycle time can slow down the DMA device by at least 33%!

Most important, our 4KRA-4 Static Memories work, and keep on working! Processor Technology has four 8800 computers, each using at least 32K Bytes of 4KRA-4 memory. We use these machines heavily for program development and product testing. We have yet to lose a single bit in normal operation! Reports from our customers confirm our experience and indicate that we have one of the lowest failure rates in the industry.

Frankly, we have done everything we could to make the best read/write memory around, because, after all, memory is the most important part of any computer.

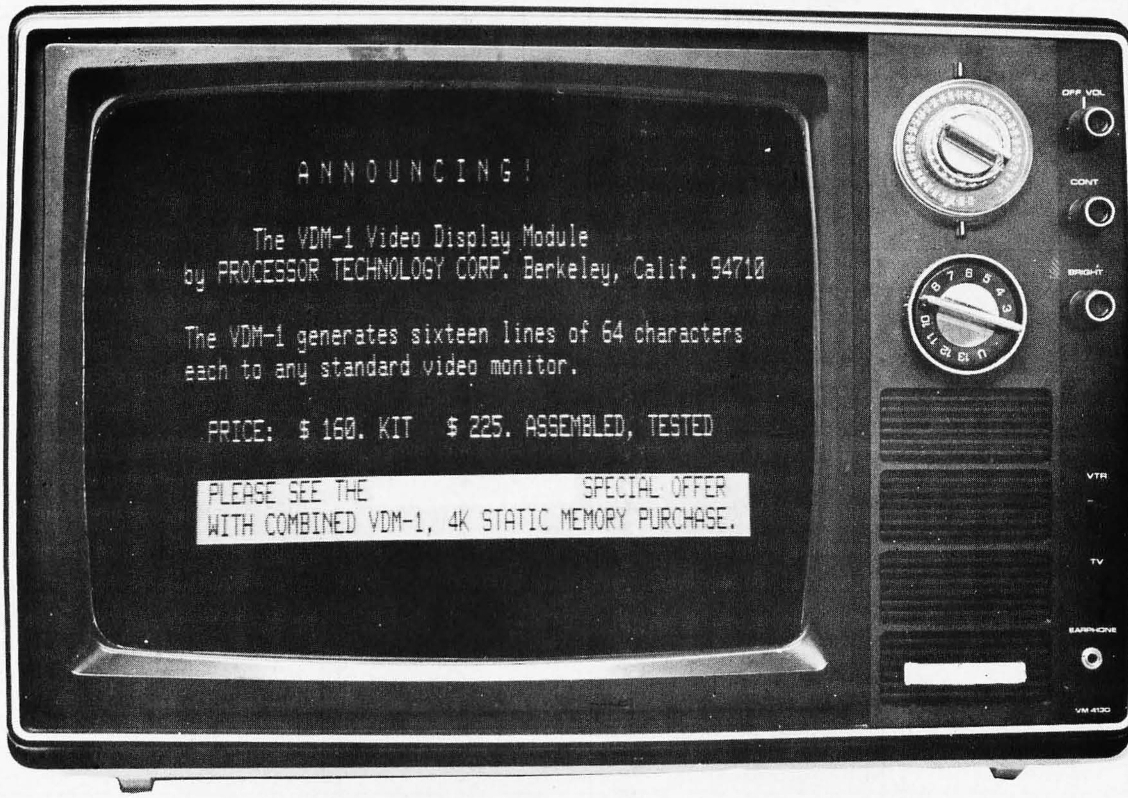
SPECIFICATIONS: 4KRA-4

Maximum capacity: 4096 eight bit bytes
Operating mode: Static
Access time: 520 nano-seconds, worst case maximum
Cycle time: 520 nano-seconds maximum, read or write
Bus Pinout: Plug-in compatible with Altair 8800 Bus
Edge contacts: Gold plated, 100 pins (dual 50) on .125" centers
Power requirements, operating: +7.5 to +10VDC at 1.0A maximum (0°C), 0.8A typical at 25°C.
standby: +1.6 to +2.5VDC at 0.5A maximum worst case, 0.4A typical
Dimensions: 5.3" x 10.0" (13.46cm x 25.4cm)

$139 KIT FORM

$195 ASSEMBLED

VDM-1
VIDEO DISPLAY MODULE



Please don't think of the VDM-1 as an ordinary TV typewriter! The VDM-1 is an intelligent display whose capabilities are limited only by your imagination! All cursor and display formations are fully programmable – there are very few hardware limitations inherrent in the design.

The VDM-1 can be used as a terminal when running BASIC or our Resident Assembler using the FREE software drivers included with every kit.

The VDM-1 contains 1024 bytes of low power high speed RAM memory which can be directly accessed by the computer as any 1K segment within its normal 65K address range. The VDM-1 is a single pc card and is plug-in compatible with the Altair 8800 bus. Multiple cursors are possible, each fully programmable. The display can be black on white, white on black, or both simultaneously. Output is standard EIA video with a signal bandwidth of 7 Mhz, compatible with any video TV monitor. The VDM-1 is so fast, efficient, and powerful we think it will soon become the standard against which other displays must be compared.

SPECIAL FREE OFFER!

Scientific Notation Software Package with Formatted Output

The floating point math package features 12 decimal digits with exponents from +127 to −127; handles assigned and unassigned numbers. With it is a 5 function calculator package: + − X & sq. root. It includes 3 storage and 3 operating memories and will handle chain and column calculations.

With the purchase of
(1) VDM-1 and (1) 4KRA-4 Memory: **$299.00** **(Offer Expires April 1, 1976)**

Please write for details on these and other Altair bus compatible modules

Processor Technology

2465 Fourth Street Berkeley CA 94710 (415) 549 0857

Continued from page 47

Table 1: Summary of CP1600 instructions. This table shows the manufacturer's mnemonics, timing requirements and comments upon operations. Instructions are grouped in several categories along the left hand edge of the table.

		MNEMONICS	OPERATION	MICROCYCLES Dir.	Indr.	Imm.	Stack	COMMENTS
External Reference Instructions	Arithmetic & Logic	ADD	ADD	10	8	8	11	
		SUB	SUBtract	10	8	8	11	
		CMP	CoMPare	10	8	8	11	Result not saved
		AND	logical AND	10	8	8	11	
		XOR	eXclusive OR	10	8	8	11	
	I/O	MVO	MoVe Out	11	9	9	9	
		MVI	MoVe In	10	8	8	11	
Internal Register Instructions	Register to Register	ADDR	ADD contents of Registers	6				Add one cycle
		SUBR	SUBtract contents of Register	6				if Register 6 or 7
		CMPR	CoMPare Registers by subtr.	6				Result not saved
		ANDR	logical AND Registers	6				
		XORR	eXclusive OR Registers	6				
		MOVR	MOVe Register	6				
	Single Register	CLRR	CLeaR Register	6				XORR with itself, except*
		TSTR	TeST Register	6				
		JR	Jump to address in Register	7*				PC ← (RRR)
		INCR	INCrement Register	6				
		DECR	DECrement Register	6				
		COMR	COMplement Register	6				One's Complement
		NEGR	NEGate Register	6				Two's Complement
		ADCR	ADd Carry Bit to Register	6				
		GSWD	Get Status WorD	6				
		NOP	No OPeration	6				Two Words
		SIN	Software INterrupt	6				Pulse to PCIT pin
		RSWD	Return Status WorD	6				
		PULR	PULl from stack to Register	11*				PULR = MVI@R6
		PSHR	PuSH Register to stack	9*				PSHR = MVO@R6
	Register Shift	SLL	Shift Logical Left	6				
		RLC	Rotate Left thru Carry	6				one or two position
		SLLC	Shift Logical Left thru Carry	6				shift capability. Add
		SLR	Shift Logical Right	6				two cycles for 2 position
		SAR	Shift Arithmetic Right	6				shift
		RRC	Rotate Right thru Carry	6				
		SARC	Shift Arithmetic Right thru Carry	6				
		SWAP	SWAP 8-bit bytes	6				2-position=SWAP twice
Control Instructions		HLT	HaLT	4				
		SDBD	Set Double Byte Data	4				Must precede external reference to double byte data
		EIS	Enable Interrupt System	4				
		DIS	Disable Interrupt System	4				
		TCI	Terminate Current Interrupt	4				Not Interruptible
		CLRC	CLeaR Carry to zero	4				
		SETC	SET Carry to one	4				
Jump Instructions		J	Jump	12				
		JE	Jump, Enable, interrupt	12				
		JD	Jump, Disable interrupt	12				
		JSR	Jump, Save Return	12				Return Address
		JSRE	Jump, Save Return & Enable	12				saved in R4, 5 or 6
		JSRD	Jump, Save Return & Disable Interrupt	12				
Conditional Branch Instructions		B	unconditional Branch	7				Displacement in PC+1
		BC, BLGE	Branch on Carry, C=1	7				PC ←PC ± Displacement
		BNC, BLLT	Branch on No Carry, C=0	7				Add 2 cycles if test condition
		BOV	Branch on OVerflow, OV=0	7				is true.
		BNOV	Branch on No OVerflow, OV=0	7				
		BPL	Branch on PLus, S=0	7				
		BMI	Branch on MInus, S=1	7				
		BZE, BEQ	Branch on ZEro or EQual	7				Z=1
		BNZE, BNEQ	Branch if Not ZEro or Not EQual	7				Z=0
		BLT	Branch if Less Than	7				S∀OV=1
		BGE	Branch if Greater than or Equal	7				S∀OV=0
		BLE	Branch if Less than or Equal	7				Z V (S∀OV)=1
		BGT	Branch if Greater Than	7				Z V (S∀OV)=0
		BUSC	Branch if Sign ≠ Carry	7				C ∀ S=1
		BESC	Branch if Sign = Carry	7				C ∀ S=0
		BEXT	Branch if External condition is True	7				4 LSB of Instruction are decoded to select 1 of 16 external conditions.

1 MICROCYCLE = 2 CLOCK CYCLES

During all arithmetic and logical operations in the CPU, the arithmetic logic unit (ALU) status bits are used to monitor and record four characteristics of the result. These bits are carry (C) out of the ALU, arithmetic overflow (OV) from the ALU, zero (Z) result from the output of the shifter, and sign (S) detect from the output of the shifter.

Applications

For small systems using only a few memory chips, limited peripheral interfacing, and/or minimal expansion capability, the CP1600 bus can be used directly without buffering. The upper bits of the 16 bit address may be used as direct chip select signals to eliminate the need for binary decoding of the upper portion of the address field. This creates a non-contiguous memory allocation but plenty of address space is available to utilize this technique.

In larger memory systems, it is best to use bipolar buffering of the bi-directional bus allowing more sophisticated peripheral interfacing and system expansion capabilities. Figure 3 shows a typical CP1600 system that provides adequate buffering of the IO bus.

The Series 1600 Microprocessor System Documentation is available from General Instrument for $20. It gives complete information on the CP1600 CPU chip including timings, programming information, and system configurations. Clearly illustrated examples are given for various features and applications including CPU start/stop control, interrupt systems, and basic input/output ports and IO interfaces. Other sections of the manual cover the available Series 1600 microcomputer modules (with complete schematics) and on-line software. The documentation package can be an especially helpful tool in building a CP1600 based system.

Conclusion

The CP1600 microprocessor provides a powerful, single-chip, CPU for the basis of a microcomputer system with minicomputer features and capabilities. Clear, concise documentation and a relatively low price tag make the General Instrument CP1600 especially appealing to the hobbyist.■

Information and diagrams courtesy General Instrument, from their Series 1600 Microprocessor System Documentation.

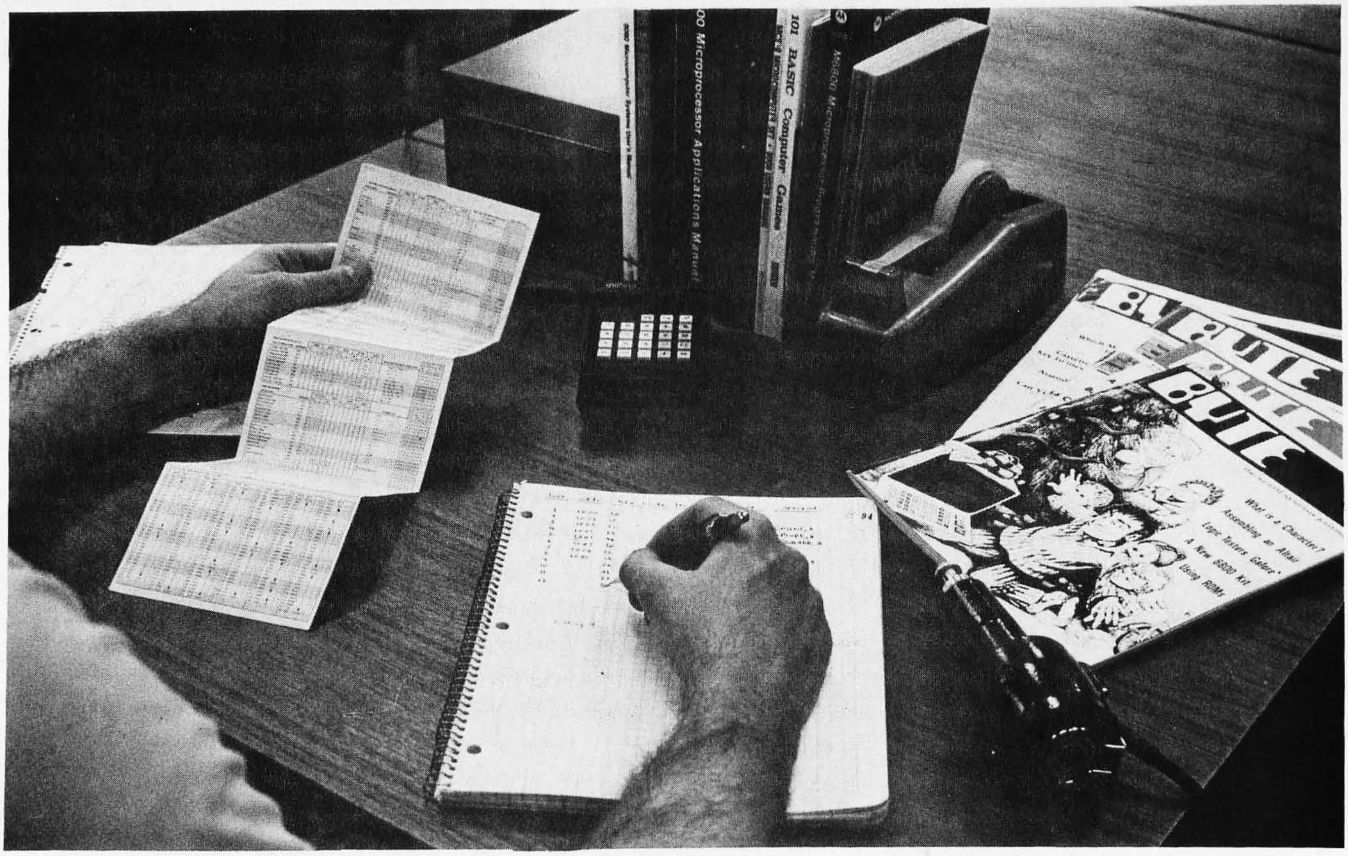


Photo 1: At the start of the hand assembly the programmer and tools might be set up as shown in this picture. Note the pencil, electric (or manual) eraser, graph paper and work table with reference volumes. A programmer's reference card is shown.

Assembling Programs by Hand

Carl Helmers

When you build your one of a kind computer from an assemblage of parts and components liberally mixed with wire wrap wire, your ultimate goal is to program the machine for experimental and practical purposes. One of the most important software elements of the complete home brew computer system is a set of programs used to help you develop further programs. This type of software programming aid is quite familiar to those of us who have a "large computer" background. The thought of a bare bones machine with no real software development tools is horrifying, to say the least. How is it possible to create this needed software generating software for the home brew computer?

One method is to implement a "quick and dirty" cross assembler by writing a set of macros for an existing computer system. The only hitch here is that you have to have access to a minicomputer (or larger computer) with the appropriate macro assembler programs. An example of this approach for an 8008 was illustrated in the article "A NOVAL Assembler," page 64, October 1975 BYTE.

Another method is to hand assemble your programs using a pencil and paper. Hand assembly is the subject of this article. The tools you'll need for hand assembly are simple: A good mechanical pencil, a matching eraser, and a work area. Supplies required are few: a number of typical graph paper sheets will suffice. A .25 inch (6.4 mm) grid will prove useful for the paper. For those readers interested in a deluxe hand assembler, a good power eraser of the type sold in architectural and drafting supply houses will work wonders; for about $35 you'll get a tool which can double as a wire wrap gun if you jury rig an appropriate bit to fit the collet intended for the eraser.

Photo 1 illustrates the setup at the beginning of a hand assembly program development task. Paper is located at a convenient position in front of the programmer, with the electric eraser nearby. A set of programming reference documentation is optional, but recommended.

The first task is to create the program and thoroughly desk debug its operation. Because it's difficult to create an assembly using this manual method, the fewer mistakes you make in programming, the less aggravating will be this painstaking task. To desk debug a program, you should go through its operation with typical examples of data and your knowledge of the computer's instruction set. The operation should be checked against what you think the program should be doing. Any errors should be corrected at this stage.

An Example of a Subroutine

As an example of a subroutine to be hand assembled, let's consider the function of moving or comparing two blocks of data in memory. This can be done by a subroutine called MBLOCK, which is called when a program needs to do the comparison or movement of data. What are the parameters required for this subroutine? Figure 1 summarizes the parameters of MBLOCK in the form of a "stack frame" which is set up for machines such as the 6800 or 8080 which can use a memory stack for data storage as well as program control. When the MBLOCK subroutine is required, each of the parameters must be set up by pushing values into the stack prior to the actual JSR instruction which calls MBLOCK. Here is a summary of what the stack frame contains, as set up by the calling sequence:

COUNT: A dummy stack element which is not initialized in the calling sequence, but is allocated for use by the subroutine.

LENGTH: The value of the length of the data field. For the purpose of this MBLOCK routine, the value of 0 for length will be treated as if 256 had been intended.

OPCODE: Designates whether the MBLOCK routine will perform a comparison (zero) or move (not zero) operation.

TOADDR: This parameter is the *address* of the block in memory which is the destination of a move, or one of the operands of a comparison.

FROMADR: This parameter is the *address* of the block in memory which is the source of data for a move, or one of the operands of a comparison.

RETURN: When the MBLOCK subroutine is reached by a JSR, the stack contains a return address as its last entry.

The information contained in figure 1 is a bit more useful than just a layout of the

Stack Frame on Entry to MBLOCK:

Symbol	Offset	Stack Content
COUNT	+8	Error count for compare
LENGTH	+7	Length of field
OPCODE	+6	Choice of move or compare
TOADDR+1	+5	Low order of destination
TOADDR	+4	High order of destination
FROMADR+1	+3	Low order of source
FROMADR	+2	High order of source
RETURN+1	+1	Low order of return address
RETURN	+0	High order of return address

Figure 1: MBLOCK uses the stack for communication with the calling program. This is the stack frame set up by the calling program and referenced by the MBLOCK program. The offsets become the address values of the symbol table entries for stack elements, and these symbols provide the first entries in the symbol table of the hand assembly.

stack. The two left hand columns form the beginnings of a symbol table for the MBLOCK program's assembly. A symbol table for an assembled routine is a list of all the labels referenced by the program, along with the address value of each label. For the parameters of MBLOCK, addresses are specified as an offset relative to the stack pointer since we will be using the stack as an argument to parameter linkage area. Whenever we use these parameters inside the detailed code of the MBLOCK routine, the 6800's indexed addressing mode will be employed after moving the stack pointer to the index register.

A typical example of the calling sequence for the MBLOCK program is illustrated in listing 1. This calling sequence is a set of 6800 instructions which are used in the example to set up the stack with values corresponding to figure 1. The typical way this is done is to load a byte into accumulator A, then push the accumulator contents onto the stack. It is not the only way the stack can be set up and the MBLOCK subroutine entered. MBLOCK does not care *how* the stack is set up, so long as it contains meaningful data when it is entered.

With figure 1 as a starting point, the MBLOCK was generated and desk debugged in symbolic assembly language, as shown in listing 2.

Graph Paper Program Representations

As mentioned earlier, graph paper figures prominently in this hand assembly process. The graph paper used as a working storage area for the hand assembly should be the typical 0.25 inch (6.4 mm) grid paper found in stationery stores. This size leaves enough

Listing 1: The calling sequence of the MBLOCK routine is used to set up the stack with arguments. The actual code used in this example is typical of such a calling sequence, but is not the only one possible.

Label	Op.	Operand	Commentary
	INS		allocate COUNT by incrementing the stack pointer;
	LDAA	#$25	pick a length for the comparison
	PSHA		or move and store it at LENGTH in the stack;
	LDAA	#1	pick the move operation code (non zero) and
	PSHA		store it at OPCODE in the stack;
	LDAA	#L(ADATA)	define the address of ADATA
	PSHA		as the destination TOADDR
	LDAA	#H(ADATA)	on the stack using immediate
	PSHA		data for both pieces;
	LDAA	#L(BDATA)	define the address of BDATA
	PSHA		as the source FROMADR
	LDAA	#H(BDATA)	on the stack using immediate
	PSHA		data for both pieces;
	JSR	MBLOCK	define the return address RETURN on the stack
			and branch to the MBLOCK routine in memory;

Notes: The symbol # indicates the immediate addressing mode for a 6800 processor: The symbol $ indicates hexadecimal notation for the number following it: H(X) indicates the high order portion of the address of symbol X: L(X) indicates the low order portion of the address of symbol X.

Listing 2: The MBLOCK program is specified here in complete detail with commentary.

Label	Op.	Operand	Commentary
MBLOCK	TSX		establish stack indexing;
	CLR	COUNT,X	clear error COUNT field;
	LDAB	OPCODE,X	fetch operation code from stack to B;
DO	LDX	FROMADR,X	address of source data to index from stack;
	LDAA	0,X	move source data byte to A;
	TSX		establish stack indexing again;
	LDX	TOADDR,X	address of destination data to index from stack;
	TSTB		is operation code in B zero?
	BNE	MOVER	if not then go to move routine;
COMPARE	CMPA	0,X	is source data equal to destination data?
	BEQ	ENDDO	if so skip to end of DO group;
	TSX		establish stack indexing;
	INC	COUNT,X	increment count of errors;
	BNE	ENDDO	if no overflow then skip special case;
	DEC	COUNT,X	if overflow then modify result to hexadecimal FF;
	BRA	ENDDO	skip around move routine;
MOVER	STAA	0,X	move source data byte to destination location;
ENDDO	TSX		establish stack indexing;
	INC	TOADDR+1,X	increment low order byte of destination address;
	BNE	NOHITO	if no overflow then skip high order incrementation;
	INC	TOADDR,X	otherwise increment high order also;
NOHITO	INC	FROMADR+1,X	increment low order byte of source address;
	BNE	NOHIFROM	if no overflow then skip high order incrementation;
	INC	FROMADR,X	otherwise increment high order also;
NOHIFROM	DEC	LENGTH,X	decrement length count by one;
	BNE	DO	if count remains then continue the loop execution;
	LDAA	COUNT,X	load final error count into A as returned value;
	LDAB	RETURN,X	move the return address up to
	STAB	RETURN+7,X	beginning of the parameter
	LDAB	RETURN+1,X	area of the stack frame while
	STAB	RETURN+8,X	preserving old stack pointer;
	LDAB	#7	modify the index register (old stack pointer)
CLEANL	INX		with this little loop to clean up
	DECB		the stack allocations for
	BNE	CLEANL	return from MBLOCK;
	TXS		redefine the stack pointer
	RTS		then return with copy acting as return pointer;

room for writing two digits or characters into each square without crowding, but the exact size is unimportant. The first step of hand assembly is to write the program down onto graph paper in the format shown in listing 3. Note that several columns are left unused at the left hand side of the graph paper. A useful format is (from left to right):

- Two columns which will receive the hexadecimal address of each instruction,
- Three columns which will receive the one, two or three bytes associated with each instruction,
- One column which will be reserved for jotting down the length of each instruction (until you get familiar enough with the instruction set to remember the lengths of each instruction);
- Spaces are used for separation.

For the purposes of this article, a typewritten copy of the program, with commentary, was presented first, in listing 2. In actual practice, the first listing you'd make of a program would be in pencil on graph paper using the model of listing 3. This would still be a full symbolic listing, but it leaves the listing in a form ready to be hand assembled. The importance of using pencil rather than pen is obvious: Unless you are an extreme perfectionist and far-sighted to boot, chances are that you will make one or two mistakes or false starts in hand assemblies of the typical programs. With a pencil and good quality bond paper, such mistakes can be erased.

Completing the Symbol Table

The first pass of a classic two-pass assembler is used to allocate the addresses of instructions or data, and to generate the symbol table of the routine being assembled. During pass one, analysis of instructions can often be used to note the numeric operation code for each line. The symbol table contains one entry for each line which has a label indicated in the label column of the source program, stored along with the address of the line on which it appeared. In an actual assembler program this address allocation process is accomplished by taking each line in turn, analyzing the operation code and operand to determine the addressing mode and hence length of the instruction, then using the derived length of the instruction to increment the location counter which keeps track of addresses. As each new line is begun, the starting address calculated from analysis of the previous line is associated with any label found in the label field and stored in the symbol table. In

Listing 3: The first step of the hand assembly process (and the last step of the program writing process) is to write a complete symbolic listing of the program on graph paper. A suggested format is shown here, with a hand written copy of the first 12 lines of MBLOCK. The secret of equal spacing is graph paper. The hand written listings in this article were made using a graph paper grid as a guide.

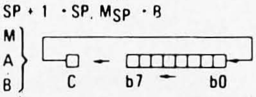
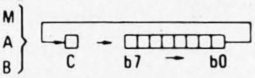
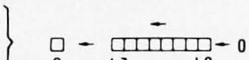
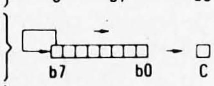
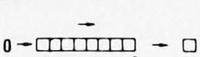
LINE	ADDR.	HEX. CODE	LEN.	LABEL	OP.	OPERAND
1				MBLOCK	TSX	
2					CLR	COUNT, X
3					LDAB	OPCODE, X
4				DO	LDX	FROMADR, X
5					LDAA	0, X
6					TSX	
7					LDX	TOADDR, X
8					TSTB	
9					BNE	MOVER
10				COMPARE	CMPA	0, X
11					BEQ	ENDDO
12					TSX	

the hand assembly process, it is often more convenient to note the lengths and operation codes of the instructions first by scanning through the program once, then to go through the program a second time assigning addresses. This makes the amount of manual thinking required at any given time smaller and thus less error prone.

It should be pointed out here that the problem of determining the length of an instruction only exists for computer designs such as the 8008, 8080, 6800, 6501 or TI 990, which have variable length instruction. If your home computer system is built around a machine with fixed word length instructions (as in some minicomputers like the PDP-8 and its microcomputer equivalent IM6100), there is no need to make any entry in the column for instruction length. In such cases, address allocation consists of jotting down an ascending sequence of numbers as addresses. However, the example being used for this article is a Motorola 6800, so the more general case of a variable instruction length applies here.

In the particular case of the 6800, the most convenient way to find the operation codes and lengths of instructions is to use the Motorola *M6800 Microprocessor Instruction Set Summary* card which is delivered along with the documentation for the 6800 computer. Figure 2 shows a copy of the side of the card which is important for hand assembly – the side which lists all the instructions. On the card itself, you will find each instruction along with its operation code in each addressing mode and its length. The important columns for this pass through the hand assembly are the columns labelled

Figure 2: The programmers' reference card supplied by Motorola for the 6800 processor. This is the side used for operation code lookups, the most common use of the card.

ACCUMULATOR AND MEMORY		ADDRESSING MODES: IMMED			DIRECT			INDEX			EXTND			IMPLIED			BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	COND. CODE REG. 5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	=	OP	~	=	OP	~	=	OP	~	=	OP	~	=		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	●	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	●	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	●	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	●	●	↕	↕	R	●
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	●	●	↕	↕	R	●
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	●	●	↕	↕	R	●
Clear	CLR							6F	7	2	7F	6	3				00 → M	●	●	R	S	R	R
	CLRA													4F	2	1	00 → A	●	●	R	S	R	R
	CLRB													5F	2	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B M	●	●	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A B	●	●	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M̄ → M	●	●	↕	↕	R	S
	COMA													43	2	1	Ā → A	●	●	↕	↕	R	S
	COMB													53	2	1	B̄ → B	●	●	↕	↕	R	S
Complement, 2's	NEG							60	7	2	70	6	3				00 M → M	●	●	↕	↕	①	②
(Negate)	NEGA													40	2	1	00 A → A	●	●	↕	↕	①	②
	NEGB													50	2	1	00 B → B	●	●	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add of BCD Characters into BCD Format	●	●	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M . 1 → M	●	●	↕	↕	4	●
	DECA													4A	2	1	A 1 → A	●	●	↕	↕	4	●
	DECB													5A	2	1	B 1 → B	●	●	↕	↕	4	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	●	●	↕	↕	R	●
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	●	●	↕	↕	⑤	●
	INCA													4C	2	1	A + 1 → A	●	●	↕	↕	⑤	●
	INCB													5C	2	1	B + 1 → B	●	●	↕	↕	⑤	●
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	●	●	↕	↕	R	●
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	4	1	A → M_{SP}, SP 1 → SP	●	●	●	●	●	●
	PSHB													37	4	1	B → M_{SP}, SP 1 → SP	●	●	●	●	●	●
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	●	●	●	●	●	●
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	●	●	●	●	●	●
Rotate Left	ROL							69	7	2	79	6	3				M	●	●	↕	↕	⑥	↕
	ROLA													49	2	1	A 	●	●	↕	↕	⑥	↕
	ROLB													59	2	1	B	●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	●	●	↕	↕	⑥	↕
	RORA													46	2	1	A 	●	●	↕	↕	⑥	↕
	RORB													56	2	1	B	●	●	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	●	●	↕	↕	⑥	↕
	ASLA													48	2	1	A 	●	●	↕	↕	⑥	↕
	ASLB													58	2	1	B	●	●	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	●	●	↕	↕	⑥	↕
	ASRA													47	2	1	A 	●	●	↕	↕	⑥	↕
	ASRB													57	2	1	B	●	●	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	●	●	R	↕	⑥	↕
	LSRA													44	2	1	A 	●	●	R	↕	⑥	↕
	LSRB													54	2	1	B	●	●	R	↕	⑥	↕
Store Acmltr	STAA				97	4	2	A7	6	2	B7	5	3				A → M	●	●	↕	↕	R	●
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B M → B	●	●	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A B → A	●	●	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A M C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B M C → B	●	●	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	●	●	↕	↕	R	●
	TBA													17	2	1	B → A	●	●	↕	↕	R	●
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M 00	●	●	↕	↕	R	R
	TSTA													4D	2	1	A 00	●	●	↕	↕	R	R
	TSTB													5D	2	1	B 00	●	●	↕	↕	R	R

INDEX REGISTER AND STACK

POINTER OPERATIONS	MNEMONIC	IMMED OP	~	#	DIRECT OP	~	#	INDEX OP	~	#	EXTND OP	~	#	IMPLIED OP	~	#	BOOLEAN/ARITHMETIC OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M + 1)$	●	●	⑦	↕	⑦	●
Decrement Index Reg	DEX													09	4	1	X − 1 → X	●	●	●	↕	●	●
Decrement Stack Pntr	DES													34	4	1	SP − 1 → SP	●	●	●	●	●	●
Increment Index Reg	INX													08	4	1	X + 1 → X	●	●	●	↕	●	●
Increment Stack Pntr	INS													31	4	1	SP + 1 → SP	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M + 1) \rightarrow X_L$	●	●	⑨	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$	●	●	⑨	↕	R	●
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M + 1)$	●	●	⑨	↕	R	●
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$	●	●	⑨	↕	R	●
Indx Reg → Stack Pntr	TXS													35	4	1	X − 1 → SP	●	●	●	●	●	●
Stack Pntr → Indx Reg	TSX													30	4	1	SP + 1 → X	●	●	●	●	●	●

JUMP AND BRANCH

OPERATIONS	MNEMONIC	RELATIVE OP	~	#	INDEX OP	~	#	EXTND OP	~	#	IMPLIED OP	~	#	BRANCH TEST	5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	4	2										C = 0	●	●	●	●	●	●
Branch If Carry Set	BCS	25	4	2										C = 1	●	●	●	●	●	●
Branch If = Zero	BEQ	27	4	2										Z = 1	●	●	●	●	●	●
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	●	●	●	●	●	●
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	●	●	●	●	●	●
Branch If Higher	BHI	22	4	2										C + Z = 0	●	●	●	●	●	●
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	●	●	●	●	●	●
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	●	●	●	●	●	●
Branch If Minus	BMI	2B	4	2										N = 1	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	4	2										V = 0	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	4	2										V = 1	●	●	●	●	●	●
Branch If Plus	BPL	2A	4	2										N = 0	●	●	●	●	●	●
Branch To Subroutine	BSR	8D	8	2										} See Special Operations	●	●	●	●	●	●
Jump	JMP				6E	4	2	7E	3	3					●	●	●	●	●	●
Jump To Subroutine	JSR				AD	8	2	BD	9	3					●	●	●	●	●	●
No Operation	NOP										02	2	1	Advances Prog. Cntr. Only	●	●	●	●	●	●
Return From Interrupt	RTI										3B	10	1		⑩					
Return From Subroutine	RTS										39	5	1	} See Special Operations	●	●	●	●	●	●
Software Interrupt	SWI										3F	12	1		●	●	●	●	●	●
Wait for Interrupt	WAI										3E	9	1		●	⑪	●	●	●	●

CONDITION CODE REGISTER

OPERATIONS	MNEMONIC	IMPLIED OP	~	#	BOOLEAN OPERATION	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0 → C	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	●	R	●	●	●	●
Clear Overflow	CLV	0A	2	1	0 → V	●	●	●	●	R	●
Set Carry	SEC	0D	2	1	1 → C	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	●	S	●	●	●	●
Set Overflow	SEV	0B	2	1	1 → V	●	●	●	●	S	●
Acmltr A → CCR	TAP	06	2	1	A → CCR	⑫					
CCR → Acmltr A	TPA	07	2	1	CCR → A	●	●	●	●	●	●

CONDITION CODE REGISTER NOTES:

(Bit set if test is true and cleared otherwise)

1 (Bit V) Test: Result = 10000000?

2 (Bit C) Test: Result = 00000000?

3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)

4 (Bit V) Test: Operand = 10000000 prior to execution?

5 (Bit V) Test: Operand = 01111111 prior to execution?

6 (Bit V) Test: Set equal to result of N⊕C after shift has occurred.

7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?

8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?

9 (Bit N) Test: Result less than zero? (Bit 15 = 1)

10 (All) Load Condition Code Register from Stack. (See Special Operations)

11 (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.

12 (All) Set according to the contents of Accumulator A.

LEGEND:

OP Operation Code (Hexadecimal).
~ Number of MPU Cycles.
= Number of Program Bytes.
+ Arithmetic Plus;
− Arithmetic Minus.
· Boolean AND.
M_{SP} Contents of memory location pointed to be Stack Pointer
+ Boolean Inclusive OR
⊕ Boolean Exclusive OR
M̄ Complement of M
→ Transfer Into.
0 Bit = Zero.
00 Byte = Zero.
H Half carry from bit 3.
I Interrupt mask
N Negative (sign bit)
Z Zero (byte)
V Overflow, 2's complement
C Carry from bit 7
R Reset Always
S Set Always
↕ Test and set if true, cleared otherwise
● Not Affected

LINE	ADDR.	HEX. CODE	LEN.	LABEL	OP.	OPERAND
1	1000	30	1	MBLOCK	TSX	
2	1001	6F	2		CLR	COUNT,X
3	1003	E6	2		LDAB	OPCODE,X
4	1005	EE	2	DO	LDX	FROMADR,X
5	1007	A6	2		LDAA	0,X
6	1009	30	1		TSX	
7	100A	EE	2		LDX	TOADDR,X
8	100C	5D	1		TSTB	
9		26	2		BNE	MOVER
10		A1	2	COMPARE	CMPA	0,X
11		27	2		BEQ	ENDDO
12		30	1		TSX	

Listing 4: This listing shows the results of an operation code and length lookup for the 12 lines of code in listing 3, and the allocation of addresses for the first 8 lines.

with "#" signs indicating the number of bytes required, and the "op" column giving the hexadecimal operation code for a particular addressing mode of the instruction.

For the Motorola 6800 design, the instruction length and operation code determination is complicated by the ambiguity of choice between the direct addressing mode and the extended addressing mode. If an operation has a possibility for direct addressing, as well as extended, the address must be known at the time the operation code is selected and the length of the instruction is determined. This can be made consistent with the hand method of assembly by requiring that all directly addressed locations be allocated in advance of assembly, in much the same way that the parameters of the MBLOCK routine were allocated relative to the stack pointer ahead of time. Since the directly addressed locations are a limited resource on the 6800, they can be treated in the same way that general registers are treated on an IBM 370 or similar machines. This means that the software would use those locations according to a specific convention created ahead of time and consistent with all the programs you plan to use in your computer. The subject of specific allocations for these locations in a 6800 is beyond the scope of this article. It suffices to note that they must be allocated ahead of time if the shorter direct addressing mode is to be used in preference to the longer extended mode, when you pick operation codes and the lengths of instructions.

Listing 4 shows how the address allocation pass proceeds once the lengths and operation codes have been noted. For hand assembly, the value of the location counter is the address of the current line being considered as it is written down. Each line starts with an address resulting from the previous line, and the object of the allocation process is to calculate the starting address for the next line. The calculation is simple: add the length of the current instruction to the starting address of the current line giving the starting address of the next line.

What about the first line of the assembled code? You will note that in listing 4, addresses start at hexadecimal 1000 at the first line. How did I pick that value? I picked the value because I happen to have some space open in my computer's programmable memory starting at location 1000; if you have programmable memory available at a different location, you would of course pick a different starting number. This process of picking a starting point is equivalent to setting the "location counter" of an assembler program. In assembler programs, the ORG pseudo operation is ordinarily used to define the initial value of a location counter, or to redefine it later during an assembly.

Listing 4 shows partial results of the operation code and length lookup, plus several lines of address allocation. At the start of the program, as noted above, the first address was chosen to be 1000 in hexadecimal. Line 1 of the listing is the 6800's TSX operation, which has an operation code of hexadecimal 30 and a length of one byte. Adding one byte to 1000 in hexadecimal gives us the allocation for the second line's starting address, 1001.

Line 2 has the 6800's CLR instruction specified with an indexed addressing mode. The notation ",X", for indexed addressing, is used throughout this example for consistency with the assembler program which Motorola uses for the examples in its manuals. The operation code 6F (hexadecimal) and length of two bytes were noted in the earlier length scan. Adding hexadecimal 2 to hexadecimal 1001 gives the starting address for the next instruction, line 3's address of 1003. This process is repeated over and over until all the addresses in the program are allocated.

While allocating the addresses, be sure to note each *symbol definition* provided by an entry in the label column of the symbolic assembly listing. When the address is calculated for a line, look to see if it has a label. Then write the label down on a separate piece of paper along with the address of the line. This separate list is the symbol table. For this routine, we start with a symbol table consisting of the symbols defined for the parameters (see figure 1) and add on the labels found when scanning through the program. After the address allocation part of pass 1, you should have a complete symbol table for your program. Table 1 shows the symbol table completed for MBLOCK.

Filling in Operand References

With the address allocation completed and summarized in a symbol table for the program, it is possible to perform by hand the second pass of the classical two pass assembler. This second pass is used to fill in all remaining address values and data values which typically depend upon symbols defined and stored in the symbol table. For a machine such as the 6800, each symbol in the table has a 16 bit value specified as a four digit hexadecimal number. During the first pass, in order to calculate the length of an instruction, its mode of addressing had to be determined as part of the selection of the operation code. The method of filling in the remainder of the instruction depends upon the addressing mode. The comments which follow refer to the 6800 design.

Immediate Addressing. The 6800 immediate addressing mode finds one or two bytes of data following the instruction byte. For instructions such as LDS and LDX, a full 16 bit address follows the instruction byte. In cases where the symbolic assembly language specifies immediate addressing with a symbol such as #MBLOCK, the second two bytes of the instruction would be filled in with the four hexadecimal digits of the address as written down in the symbol table. For instructions such as the LDAB found on line 32 of listing 2, the immediate value is a one byte quantity since the operation involves the normal data length for the machine.

Direct Addressing. For 6800 operations which use direct addressing, the second byte of these two byte instructions is simply the low order portion (rightmost two digits) of the value of the hexadecimal address written down in your symbol table as in table 1.

Extended Addressing. For 6800 operations which use extended addressing, the second two bytes of the instruction contain the full 4 digit hexadecimal (16 digit binary) representation of the address as noted in your symbol table. Thus, for example, the JSR at the end of the example in listing 1 would be assembled with a 4 digit hexadecimal address of 1000 forming the second two bytes of the instruction. (One reason the 6800 is easier to assemble by hand than either the 8008 or the 8080 is that its many extended addressing modes use the two bytes of the address in the natural order of the hexadecimal number.)

LINE	ADDR.	HEX. CODE		LEN.	LABEL	OP.	OPERAND
1	1000	30		1	MBLOCK	TSX	
2	1001	6F	08	2		CLR	COUNT, X
3	1003	E6	06	2		LDAB	OPCODE, X
4	1005	EE	02	2	DO	LDX	FROMADR, X
5	1007	A6	00	2		LDAA	0, X
6	1009	30		1		TSX	
7	100A	EE	04	2		LDX	TOADDR, X
8	100C	5D		1		TSTB	
9	100D	26	0D	2		BNE	MOVER
10	100F	A1	0D	2	COMPARE	CMPA	0, X
11	1011	27	0B	2		BEQ	ENDDO
12	1013	30		1		TSX	
13	1014	6C	08	2		INC	COUNT, X
14	1016	26	06	2		BNE	ENDDO
15	1018	6A	08	2		DEC	COUNT, X
16	101A	20	02	2		BRA	ENDDO
17	101C	A7	00	2	MOVER	STAA	0, X
18	101E	30		1	ENDDO	TSX	
19	101F	6C	05	2		INC	TOADDR+1, X
20	1021	26	02	2		BNE	NOHITO
21	1023	6C	04	2		INC	TOADDR, X
22	1025	6C	03	2	NOHITO	INC	FROMADR+1, X
23	1027	26	02	2		BNE	NOHIFROM
24	1029	6C	02	2		INC	FROMADR, X
25	102B	6A	07	2	NOHIFROM	DEC	LENGTH, X
26	102D	26	D6	2		BNE	DO
27	102F	A6	08	2		LDAA	COUNT, X
28	1031	E6	00	2		LDAB	RETURN, X
29	1033	E7	07	2		STAB	RETURN+7, X
30	1035	E6	01	2		LDAB	RETURN+1, X
31	1037	E7	08	2		STAB	RETURN+1+7, X
32	1039	C6	07	2		LDAB	#7
33	103B	08		1	CLEANL	INX	
34	103C	5A		1		DECB	
35	103D	26	FC	2		BNE	CLEANL
36	103F	35		1		TXS	
37	1040	39		1		RTS	

Listing 5: The final result of hand assembly is a complete listing of the program with all addresses allocated, and all hexadecimal codes for instruction bytes completed. This listing shows the result for MBLOCK. The next step is of course to load the program and test it in your computer.

Indexed Addressing. The example of MBLOCK uses indexed addressing almost exclusively for data manipulation since all the variable data is placed in the stack to make the routine reentrant. In indexed addressing, the second byte of the instruction contains a positive offset from the value currently in the index register. To fill in the second byte of such instructions, you use the two low order digits of the hexadecimal value of the symbol as noted in your symbol table. Thus for the instruction of line 15 of MBLOCK, the value noted down in the second byte of the instruction at this stage of the assembly is hexadecimal '08' found by looking up COUNT in the

Table 1: Symbol Table for the MBLOCK program. This table lists the symbol and its value for each symbol referenced in MBLOCK.

Symbol	Hexadecimal Value
COUNT	0008
LENGTH	0007
OPCODE	0006
TOADDR	0004
FROMADR	0002
RETURN	0000
MBLOCK	1000
DOFOR	1005
COMPARE	100F
MOVER	101C
ENDDO	101E
NOHITO	1025
NOHIFROM	102B
CLEANL	103B

symbol table. (The ",X" as noted earlier merely identifies indexed addressing.) Note that if you have a symbol table value greater than hexadecimal FF, this rule would cause a truncation and possibly erroneous operation of your program. Indexed references can not have offsets greater than hexadecimal FF in the 6800 architecture.

Inherent Addressing. For instructions with inherent addressing, there is no more to do once the operation code is noted. Such instructions always address some particular facet of the CPU's architecture and as a result have no optional addressing to be determined during the assembly process. Examples include the RTS of line 37 as well as the TXS instruction occurring at several places in MBLOCK.

Relative Addressing. The one remaining 6800 addressing mode is the relative addressing mode. This is also the most difficult mode to handle in a hand assembly for this machine, because the instruction requires calculation of a branch address relative to the current instruction's location in memory. This calculation requires that you do hexadecimal arithmetic. First let's consider the case of a forward reference. A forward reference occurs when the address of the symbol mentioned in the branch is located at a higher address than the current address. An example is the forward reference at line 9 of MBLOCK to the label MOVER found on line 17 of MBLOCK. The symbol table comes in handy here. We look up the symbol of the target of the branch, MOVER. Then we subtract the address of the next instruction following the branch. In the example, the address of the instruction on line 10 is 100F, and the address of MOVER is 101C so the difference becomes 101C − 100F = D. The low order two digits of this hexadecimal number can be used for the branch offset provided the number is less than or equal to hexadecimal 007F. (Relative addressing outside this limit cannot be done on the 6800, so branching to a nearby JMP instruction with extended addressing may be required. Here is a case where pencil and eraser become necessary, since you won't know your branch is out of range until after the address allocation and hexadecimal subtraction are performed.)

Next, consider a backward reference. A backward reference is a reference to a previously defined label. An example in MBLOCK is the conditional branch to DO at line 26. Here, as in the previous case, the branch offset is calculated as the difference of the target address minus the next instruction address. But since the next instruction address exceeds the branch target address, the result will be negative. In the example of the branch to DO, the branch is at location 102D so that location 1005 minus location 102D gives a result of FFD8. This number must be interpreted as an 8 bit two's complement negative number in order to have the proper hexadecimal code. Its magnitude as a full four digit hexadecimal number must exceed FF80 for a proper 8 bit offset to be taken from the low order digits.

Before leaving the subject of hand assembly and the 6800's relative addressing mode, there is one trick which will prove quite useful for generating the offset field of short relative jumps: Counting. Simply consider the second byte of the instruction as residing at a count of hexadecimal FF. To figure out a forward reference offset, count bytes forward in hexadecimal. The next number after FF in an 8 bit counting sequence is 00. Thus to go to the starting address of the instruction following the branch, use 00. Simply counting bytes forward will give the offset for a branch target at any positive displacement. Counting backward from FF in hexadecimal gives the same effect for backward references. To branch to itself, a branch instruction has FF for its second byte, and counting backward has FE for its first byte. For earlier instructions, keep the sequence going: FD, FC, FB . . . until you get to the first byte of the desired branch target. (Stop at 80, however, since that represents the maximum displacement backwards since 7F is a positive displacement.)

What Next?

The result of carrying out the hand assembly process is presented in complete detail for MBLOCK in listing 5. This listing contains a reproduction of the complete hand assembly of MBLOCK with an absolute address origin of 1000. When you reach the stage of listing 5, your assembly is complete, and the program is ready to test. Load the program starting at the address of the first *line of code*, using whatever means is appropriate for your computer. For Altair 680 users with a minimal system, this would be via panel switches. For SWTPC 6800 system users, this would be via the MIKBUG program software using a terminal such as a TVT or a Teletype. Sphere system owners could use their miniature assembler program to eliminate much of the drudgery of address calculation for relative branches, but handwritten listings to document programs would still require many hand assembly elements; the results of a Sphere assembly are of course already in memory, so loading is not required. Hopefully, at some bright point in your computer's future, you'll have enough memory and there will be software products available, so that you can do a full assembly by machine, using cassettes for storage media and perhaps a Teletype for hard copy – at which point you can pack up your pencil and electric eraser.■

Processing Algebraic

In an article which appeared last month, we showed how the small system user can process algebraic expressions by using the Bauer-Samelson algorithm, developed by F L Bauer and K Samelson at the Technische Hochschule in Munich, Germany. The Bauer-Samelson algorithm has many variations, depending on the type of algebraic expression processing we wish to do. The most interesting of these have to do with the process of compiling.

Anyone who has ever thought about writing a compiler has probably already guessed that there are certain aspects to compiler writing that are not hard at all. Consider, for example, GO TO statements. If a compiler is reading, as input, a source program, and it comes to the words GO TO, it proceeds in a very simple manner. It reads the next few characters – let us say they are 305, so that the statement is GO TO 305 – and it writes, as output, whatever the machine code is for a transfer to some point in the object program corresponding to the label 305. The only part of this that is difficult at all is keeping a table of labels (such as 305) and their corresponding addresses. That can be somewhat complex, especially when GO TO 305 is a so-called *forward reference* – that is, when it *precedes* the label 305 in the source program.

W Douglas Maurer
University Library Room 634
George Washington University
Washington DC 20052

A Single Register Machine

But all that pales into insignificance beside the problem of compiling code for algebraic expressions. Let us consider the simplest possible case. We have a machine with one register ("the accumulator") and six instructions as follows:

LDA	Load Accumulator
ADD	Add to Accumulator
SUB	Subtract from Accumulator
MPS	Multiply Single Register
DVS	Divide Single Register
STO	Store Accumulator

(We are assuming for the moment that our multiply instruction produces a single register result in the accumulator, and that our divide instruction divides a single register quantity in the accumulator by a quantity in memory. This restriction will be relaxed later on.)

Suppose now that we have an assignment statement such as

$$K = (I*J - I + J)/N$$

If this is read by the compiler as part of the source program, then the compiler must write out the equivalent machine language or assembly language code, which in this case would be

```
LDA   I
MPS   J
SUB   I
ADD   J
DVS   N
STO   K
```

or the corresponding machine language (absolute binary or octal or hexadecimal) code. How is this code to be produced?

Expressions Part 2

Once you know how to do basic processing on algebraic expressions, you can begin to learn how to write compilers.

We will now describe a modification of the Bauer-Samelson algorithm that produces such code. The main areas of modification are as follows:

(1) The "result" of a computation, which is calculated by the unstacking process, is no longer a number, but rather a *place* where the result of the computation is stored. For all of the instructions above (except STO), this will be the accumulator. Thus a special code to signify the accumulator will be placed on the operand stack.

(2) Every time unstacking takes place, output code is generated in addition to the calculation of the result.

Let us go through the above assignment statement as an example. (Refer to BYTE No. 6 for a general discussion of the Bauer-Samelson algorithm.)

(1) The left parenthesis goes on the operator stack.

(2) The I goes on the operand stack. (In this version of the Bauer-Samelson algorithm, we put *variables* – or pointers to them – on the operand stack, and not their values.)

(3) The * goes on the operator stack.

(4) The J goes on the operand stack.

(5) Now we cannot put the (binary) minus sign on the operator stack, because it has lower precedence than the * operator. So we must unstack the *. We take it off the operator stack, and its operands, I and J, off the operand stack; and now we must calculate a result.

Since I and J are the operands and * is the operator, we must generate code to multiply I by J. The code that does this is

```
LDA    I
MPS    J
```

or its machine language equivalent as above; and the result, I*J, is left, by these two instructions, in the accumulator. Let us denote the accumulator by $AC (the $ is there so that we cannot possibly confuse this with the name of a variable in the program, such as AC); then $AC goes on the operand stack. Now we can put the minus sign on the operator stack, directly above the left parenthesis.

(6) The second I goes on the operand stack.

(7) We cannot put + on the operator stack, because its precedence is equal to that of the minus sign, which we must now unstack. We take it off the operator stack, and we take its operands, I and $AC, off the operand stack. Remember that the *second* operand is taken off first; so the operands are actually $AC and I. What instruction performs the subtraction $AC – I? Clearly

```
SUB    I
```

is the one we want. (If the subtraction were I – $AC, this would have to be followed by another instruction which complements the value in the accumulator.) So the above instruction is generated; and, since it leaves its result in the accumulator, $AC is put back on the operand stack. Now we can put

An *interpreter* analyzes algebraic expressions every time a calculation is made; by carrying the process one step further we get a *compiler*, which analyzes the expression once while creating a specialized machine language program to do the calculations.

The Bauer-Samelson algorithm will always generate code from left to right. The result will not necessarily be as fast as optimal code by a human programmer.

There is often the problem of data types: If all data is in the form of n bit integers, this is not a problem; but when multiple types of data are allowed, mechanisms for conversion are required.

+ on the operator stack, directly above the left parenthesis.

(8) The second J goes on the operand stack.

(9) Now we come to the right parenthesis. This means that we must unstack the + on the operator stack. Its operands are $AC and J (after reversing the order, as above); so, just as in step 7, we want to generate the instruction

```
ADD    J
```

and put its result register, namely $AC, back on the operand stack. Now the operator at the top of the operator stack is a left parenthesis; this is removed, leaving the operator stack empty.

(10) The / goes on the operator stack.

(11) The N goes on the operand stack.

(12) We are now at the end of the expression, and we must unstack the / and generate the instruction

```
DVS    N
```

This is done in the same way as in steps 7 and 9, leaving the operator stack empty and $AC on the operand stack.

(13) Finally – and this is not, strictly speaking, part of the Bauer-Samelson algorithm – we look at the left side of the = for the first time, namely K, and generate the instruction

```
STO    K
```

to complete the generation of code in this case.

We have purposely picked a rather easy example, involving no temporary variables, no quotient register, and so on. This is by no means all there is to this version of the Bauer-Samelson algorithm, but the further refinements are not hard to visualize.

First of all, we must make sure that we can generate code for all possible cases. For a reason which will become apparent, the special symbol $AC will never be on the operand stack in two different places. So the operands of any given operator will always be in one of the following three forms: $AC and Y, Y and $AC, or X and Y. All of these cases have been treated, or at least mentioned, above. The first two cases are, of course, equivalent if the operator is + or * (since $AC + Y = Y + $AC and $AC * Y = Y * $AC).

The second case above, in which we may have to use more than one instruction (subtract followed by complement, for example, as discussed above) corresponds to the case in which a human being might generate different code from that generated by the algorithm. Suppose, for example, that our expression is A*D – B*C. A human being would generate the code to multiply B and C *first*, and later subtract it from A*D. The Bauer-Samelson algorithm, however, will always generate code from left to right. Ultimately, it will calculate B*C – A*D and then complement this, producing A*D – B*C. The resulting code will not, of course, be as fast as the code that a human being would generate. However, the difference in speed is minimal, and the so-called "optimization techniques" which allow computers to produce better code are probably too bulky to fit into your small system.

The above example expression, A*D – B*C, illustrates two further problems with compiling of expressions. The first is that of data types. If we use the FORTRAN conventions, A, B, C, and D are all real numbers, and we have to use floating point addition, subtraction, multiplication, and division. In many small systems, there are no real numbers, but the problem of data types may still remain. There may be 16 bit and 32 bit integers, signed and unsigned integers, and so on, each of which has its own instruction set. If mixed mode expressions are not allowed, we may determine the type of an expression as soon as we see the first variable in it, and make sure that we use only addition, subtraction, multiplication, and division instructions of that type.

(If we *do* allow mixed mode expressions, then it will be necessary to put a code – $REAL, $INT16, $INT32, or the like – on the operand stack along with *each* quantity placed there to record the type of that quantity. When we unstack, we must now generate code to add, subtract, multiply, or divide two quantities of the types given, and calculate not only the result – which we put back on the operand stack – but also its type. Thus all quantities on the operand stack, in that case, are pairs, each of which consists of a variable, register, etc., together with its data type.)

The second problem illustrated by A*D – B*C is the use of temporary variables. In our one register machine, we will have to store the value of A*D (or B*C) in a temporary location during the calculation. This store instruction is generated when we are trying to load a register – in this case the accumulator – whose contents cannot be destroyed, as evidenced by the fact that the symbol for this register is currently on the operand stack. To illustrate this process, we shall go through the above example in the same way as we did before. The code we will generate

for the evaluation of the expression A*D − B*C is

```
LDA   A
MPS   D
STO   TEMP1
LDA   B
MPS   C
SUB   TEMP1
COM
```

(where COM stands for "complement the value in the accumulator"), and this is generated as follows:

(1) A goes on the operand stack.
(2) * goes on the operator stack.
(3) D goes on the operand stack.
(4) We cannot put − on the operator stack, since it has lower precedence than *, which we must therefore unstack. We take * off the operator stack and A and D off the operand stack, and generate code to multiply A by D, just as before, that is,

```
LDA   A
MPS   D
```

Since the answer is left in the accumulator, we put $AC on the operand stack. At the same time we keep a pointer to this stack position in a special cell which we shall call ACSP (for $AC Stack Position). In this case, the pointer value is 1, since $AC is the *first* quantity on the operand stack (counting from the bottom). ACSP is initialized to zero, and whenever it is zero, it is assumed that $AC is *not* currently on the operand stack.

Now we can proceed to put − on the operator stack, which was left empty by the previous unstacking.

(5) B goes on the operand stack.
(6) * goes on the operator stack (since its precedence is higher than that of the operator at the top of that stack, namely −).
(7) C goes on the operand stack. We are now at the end of the expression and must unstack all the operators on the operator stack.
(8) First we unstack the *. Its operands are B and C, and it would seem that the code we should generate is

```
LDA   B
MPS   C
```

However, there is a problem. If we generate the first of these two instructions, we will be loading the accumulator and destroying its current contents, which we need. We can tell that we need the current contents of the accumulator because ACSP is not zero. In fact, the quantity currently in the accumulator is the value of A*D.

Therefore the rule is as follows: Whenever we are about to generate a *load* instruction, we first check to see if ACSP is zero. If it is, we may proceed. If it is not, however, we must generate another instruction to store the accumulator into a temporary cell − in this case, TEMP1. The name TEMP1 is now put on the operand stack *in place of* $AC. It is *not* (necessarily) put on the top of that stack. Instead, we look at the pointer to see where to place it. In this case, the pointer value is 1, so that TEMP1 becomes the *first* element on the operand stack (counting from the bottom), which is where $AC was before. At the same time, ACSP must be set to zero, denoting the fact that $AC is no longer on the operand stack.

Evaluation of expressions on a single register machine often requires use of temporary operands in memory.

(In some algebraic expression evaluations, we will need more than one temporary cell. Let us call these TEMP1, TEMP2, TEMP3, etc. We have a temporary cell counter which is initialized to zero. Every time we need a new temporary cell, as above, we may increase this counter by 1. Alternatively, we may check the operand stack to see what temporary cells are currently on it, and pick out a new one in this way. If a *new* temporary cell is needed, it cannot be currently on the operand stack; however, its choice is otherwise unrestricted.)

Let us assume, therefore, that we have generated

```
STO   TEMP1
```

followed by the two instructions as above. The result of these two instructions is left in the accumulator, so $AC goes back on the operand stack. Note that the contents of the operand stack were $AC, B, and C, with C on top; then $AC was changed to TEMP1 and B and C were taken off. Now with $AC put back on, the contents of this stack are TEMP1 and $AC.

(9) Now we unstack the −. Its operands, as given above, are TEMP1 and $AC. If the − were a + we could simply generate

```
ADD   TEMP1
```

and we would be done. However, as it stands, we have a problem, because simply generating

```
SUB   TEMP1
```

would perform the operation $AC − TEMP1, rather than TEMP1 − $AC. We may

Generation of code for expressions evaluated on multi-register machines can use the extra registers as temporary storage; however, this introduces the need to keep track of register usage and special cases.

notice that \$AC – TEMP1 = –(TEMP1 – \$AC), and, therefore, the instructions

```
SUB     TEMP1
COM
```

will perform the subtraction we want.

Multi-register Machines

The use of the special cell ACSP as above is a special case of the use of a *table of register contents*. In general, a computer will have more than one register. For *each* such register (that participates in the instructions to be generated), we will need a location like ACSP. All these locations are initialized to zero at the start of evaluation; each time one of them is used, it will appear on the operand stack, and a pointer to its position there will be kept in the location corresponding to that particular register. In some cases we may simplify matters and keep only Boolean values (zero or one) in these locations; we can always search the operand stack, if we have to, to find where a register is on that stack, if the corresponding Boolean value is 1.

As an example, suppose that we have a more typical kind of multiplication instruction which leaves a double word answer in the accumulator and a *quotient register*, which we shall denote by \$Q on the operand stack. If the quantities being multiplied are integers and the quotient register is the least significant part of the double register result, we can assume that the result goes in the quotient register and put \$Q on the operand stack immediately after generating a multiply instruction. This in turn means that we have to be prepared to accept \$Q as an operand. If X and \$Q, for example, are the operands of + (which is being unstacked) and there is no instruction to add X to the quotient register directly, there may be an instruction, which we can generate, to move the contents of the quotient register to the accumulator (or to exchange these two registers), after which we can generate an instruction to add X in the normal way. Of course, in this case, we have to check the location corresponding to \$Q before we generate a multiply, to see whether the quotient register has to be stored in a temporary location.

In a multi-register machine, we will not usually need any temporary cells. (By a multi-register machine, we mean here, specifically, a machine with more than one *arithmetic* register, in which addition, subtraction, multiplication, and division can take place.) All we need is to make sure, whenever we load a new register, that this register is not already being used for some other purpose. Let us illustrate this by considering again the expression A*D – B*C. The code generated for this, on a multi-register machine, would be roughly as follows:

(1) Load some register U with A.

(2) Multiply D by the contents of register U.

(3) Load some other register V with B.

(4) Multiply C by the contents of register V.

(5) Subtract one register from the other.

Here the registers U and V will have corresponding stack position variables, which for the moment we shall call USP and VSP. At the beginning, these are set to zero. When it comes time to generate the first two instructions ("load A" and "multiply by D"), we search for a register in which this computation can be performed. We find that it can be performed in U (because USP = 0), and so we generate instructions which make use of the register U. At the same time, we set USP to indicate that register U is now in use (assuming that the multiply instruction leaves its answer in U). Now, when it comes time to generate the next two instructions ("load B" and "multiply by C"), we again look for a register that we can use. This time we determine that we cannot use U (because USP ¬= 0); so we have to keep on looking. If V is the next register that we look at, we see that we can use it, since VSP = 0, and so we generate the third and fourth instructions above in such a way that they make use of the register V.

Stack Machines and Lukasiewicz Notation

Besides conventional single register and multi-register machines, there are *stack machines*. A load instruction on a stack machine puts the quantity to be loaded on the top of a stack of registers; a store instruction removes (pops up) the quantity to be stored from the top of this stack. An add instruction removes two registers from the top of the stack, adds their contents, and puts the result back on top of the stack. Thus a stack machine effectively performs about half the Bauer-Samelson algorithm in

hardware, and the generation of code for such a machine is considerably simplified. We shall now describe how this is done.

The code for calculation of an algebraic expression on a stack machine is directly related to the form of that algebraic expression expressed in *Polish notation*. (The proper name for this is Lukasiewicz notation, but it is popularly called Polish notation because very few people can pronounce Lukasiewicz – an English approximation, however poor, is WOO-kah-SHEV-itch. Other names for Polish notation are "suffix notation" and "reverse Polish." There is also "prefix notation" or "forward Polish," but this is never used in this context in computing.)

The Polish notation equivalent of a one operator expression, such as A+B or C–D, is formed by taking out that operator and putting it at the end: A B + or C D –. The Polish notation equivalent of a more complex expression is formed by breaking it down into parts, normally two parts with an operator between them; this operator is placed at the end, and the two parts are themselves expressed in Polish notation. Thus for (A+B)*(C–D), the two parts are A+B and C–D, or, in Polish notation, A B + and C D –, and the operator is *, so the entire expression is A B + C D – *. Similarly, A*D – B*C is expressed in Polish notation as A D * B C * –.

Conversion of an algebraic expression in ordinary, non-Polish (or, as it is often called, *infix*) notation into Polish notation may be performed by using a drastically simplified version of the Bauer-Samelson algorithm. There is only one stack, namely the operator stack. Operands, instead of being put on a stack, are put directly on the end of the string in Polish notation which is being constructed. As an example, let us go through the string A*D – B*C once again, according to this version of the algorithm:

(1) A goes on the end of the string.

(2) * goes on the stack.

(3) D goes on the end of the string, which is now A D.

(4) * is unstacked, that is, placed on the end of the string, which is now A D *.

(5) – goes on the stack.

(6) B goes on the end of the string, which is now A D * B.

(7) * goes on the stack (as before, since its precedence is greater than that of –).

(8) C goes on the end of the string, which is now A D * B C.

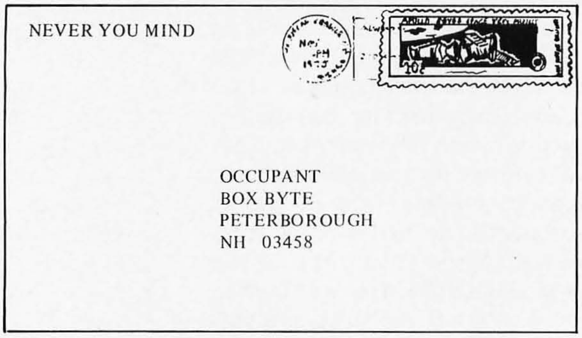
(9) * is unstacked, so the string is now A D * B C *.

(10) – is unstacked, so the string is finally A D * B C * –.

Once a string in Polish notation (or "Polish string") has been formed, code to calculate the value of the corresponding algebraic expression may be generated directly. Each operand corresponds to a load instruction, and each operator corresponds to an instruction which implements it. Thus in the above case the instructions would be: Load A; load D; multiply; load B; load C; multiply; subtract. The first two of these instructions load A and D onto the register stack in our stack machine. The next instruction takes A and D off the stack and puts A*D back on. The next two instructions put B and C on the stack of registers, which now contains A*D, B, and C. The next multiply instruction takes B and C off the stack and puts B*C back on; the final subtract instruction takes A*D and B*C off the stack and puts A*D – B*C back on. After this code, we can have an instruction to store the result, and this instruction leaves the register stack the way it was before expression evaluation started.

A stack machine effectively performs half the Bauer-Samelson algorithm in hardware, so code generation is considerably simplified.

Polish notation is also often used in *interpreters*. An interpreter is like a compiler, except that no code is generated; instead, the interpreter actually performs the indicated instructions as it goes. Typically, an interpreter will go through an initial phase (often called, confusingly, a "compiler phase") in which the program to be interpreted is read, and all expressions converted to Polish notation (among other things) and stored internally in this way. The actual interpretation now follows, with the interpreter moving from one statement of the interpreted program to the next, doing what each statement says and proceeding to the interpretation of whichever statement comes next in logical order. (Thus if it is interpreting IF K=0 THEN GO TO ALPHA, and K is in fact zero, then the statement labeled ALPHA will be interpreted next.) The advantage of keeping expressions in an internal form corresponding to Polish notation, rather than ordinary infix notation, is that Polish notation may be evaluated much more efficiently than infix notation. All we have to do to evaluate a Polish string is to simulate the effect of a stack machine, as outlined in the preceding paragraph. That is, we go through the Polish string from left to right; whenever we come to an operand, we place it on a stack, and whenever we come to an operator, we act as if we were unstacking it. This is the "other half" of the Bauer-Samelson algorithm; like the algorithm given above to convert a string into Polish notation, it uses only one stack, but this is an operand stack rather than an operator stack.■



FILLING A VACUUM WITH PROMs

The answer to the article on page 12 (BYTE December 1975) by Chris Ryland concerning proprietary programs and their duplication via IBM, Xerox or similar copiers, may be PROMs.

Why don't vendors put their programs in PROMs (or ROMs) and sell them as "hardware"? The programs could be sold to reside in several different storage areas and use customer RAM for fields that need to be written. The users would not get listings, only function descriptions of what the program does, where it is in memory, and where (and how much) RAM it needs. For example consider the following prototype advertisement.

> FOR SALE: Widgit Program. Displays 1 to 47 hexagonal widgits in a 512 byte RAM buffer. Buffer may be displayed on a CRT as 16 lines of 32 characters per line. Available from stock for PROM locations 8 K, 32 K and 60 K; stock RAM locations 0 K or 4 K; length 539 bytes. Custom address allocations available at extra charge.

Why don't vendors put their programs in PROMs (or ROMs) and sell them as "hardware"?

Of course the details would change depending upon what the product is and how important it is.

DP Parks
Cary NC

MORE ON VACUUMS

You are doing an excellent job with BYTE. Each month, when a new issue arrives, I attack it with the "total immersion method" and don't come up for air until every article is finished. I am certain that the clearing house and tutorial functions BYTE performs are greatly enhancing both the pleasure of computer hobbyists and the speed with which home computing is developing.

I have for some time been thinking about the software requirements of both home and hobbyist computing. I thus found your December editorial and the complementary article on "The Software Vacuum" by Chris Ryland particularly interesting.

There are several factors which are certain to govern the development of hobbyist and home computing in the years to come:

1. There will continue to be dramatic improvements in hardware performance including processor speed, instruction set sophistication, and memory speed and capability.
2. There will continue to be a proliferation of non-compatible but new, powerful and otherwise desirable CPU chips.
3. The price to performance ratio of hardware will continue to improve.
4. Software development will continue to be labor intensive and therefore expensive.
5. Application programs will continue to be limited in sophistication, and computers will continue to be exotic toys of only arcane interest to a minority *unless* powerful, easy to learn and use high level languages and other software are readily available.
6. The amount of computer power per dollar available from manufacturers, the number of competitive and complementary systems on the market, and the quantity and quality of services available from both hardware and software suppliers are all directly related to the size of the market. That is, the more hobbyists with compatible systems, the better off all of us will be.

Thus the quality and power of hardware we will be able to afford in the future is dependent on the desirability and transferability of the available software. As Mr. Ryland pointed out, software is not going to "grow on trees." Complex packages may be developed by amateurs, but their maintenance is often a full time job. Since copying costs are relatively low, it is a difficult problem to create ways by which the software supplier can assure himself a return on his investment.

I believe the problem is somewhat comparable to the sale of recorded music. In that business there is a problem of "record pirates," but most companies make a reasonable profit. They succeed because the market is large enough to allow the unit



Now, you can buy an Altair 8800 or Altair 680 computer kit right off the shelf. Most all Altair options, software and manuals are also available. The MITS Dealer List below is just the beginning:

off the shelf

ARROWHEAD COMPUTER CO.
(The Computer Store)
11[illegible]56 W. Pico Blvd.
Los Angeles, CA 90064
(213) 478-3168

BYTE'TRONICS
5604 Kingston Pike
Knoxville, TN 37919
(615) 588-8971

THE COMPUTER SYSTEMCENTER
3330 Piedmont Road
Atlanta, GA 30305
(404) 231-1691

COMPUTER KITS
1044 University Ave.
Berkeley, CA 94710
(415) 845-5300

COMPUTERS & STUFF
1092 S. State St.
Orem, UT 84057
(801) 224-2066

GATEWAY ELECTRONICS
2839 W. 44th Ave.
Denver, CO 80211
(303) 458-5444

GATEWAY ELECTRONICS
8123-25 Page Blvd.
St. Louis, MO 63130
(314) 427-6116

MARSH DATA SYSTEMS
5405-B Southern Comfort Blvd.
Tampa, FL 33614
(813) 886-98[illegible]0

MICROSYSTEMS
6605A Backlick Rd.
Springfield, V[illegible] 22150
(Washington DC area)
(703) 569-1110

CTI DATA SYSTEMS
3450 East Sprin[illegible] St.
Long Beach, CA [illegible]0806
(213) 426-737[illegible]

BYTE SHOP
1063 El Camino Real
Mountain View, [illegible]A 94043
(415) 969-5464

RIDGWAY EAST
Retail Computer Center
161 Bell St.
Chagrin Falls, OH 44022
(Cleveland) (216) 247-4845

THE COMPUTER STORE, INC.
P[illegible] Box 26[illegible]1
Fr[illegible]mingham Cent[illegible]r, MA 01701
(6[illegible]7) 877-[illegible]984

NOTE: Altair is a registered trademark of MITS, Inc.

return to be small in comparison to copying costs. If this analogy is correct, we may be in the unenviable position of not having a large market because of the software vacuum until we have a large market. [The old bootstrapping problem.]

In spite of this problem I believe there are at least three likely sources of software. First, the CPU chip manufacturers will continue to provide software. This software will be relatively inexpensive because they will expect it to increase sales of both the CPU chips and memory. As the amount of memory in the average microcomputer grows, the chip manufacturers will probably move toward higher level languages and other sophisticated software. The chip manufacturers will also probably sell more and more software in mask programmed read only memory as the market develops. Second, microcomputer manufacturers will continue to be a source of software of increasing complexity. Again, their primary goal will be to increase the desirability of their product. Unfortunately, it will be very difficult for them to avoid software pirating by users, and the use of their software on competitive systems. If these problems are excessive we will see less software from this source. Third, we can expect to see "surplus software" on the market. Many OEMs [original equipment manufacturers] who use microcomputers in their products are bound to develop software for their own use. Over a period of time, this software will either enter the public domain on a non-supported basis, or will be sold as a sideline of such companies.

One solution might be to identify the 256 most valuable and powerful instructions we might desire in an "ideal" 8 bit processor.

Unfortunately, all of the above sources are only going to help users of the specific chip to which they are directed. It is almost as if, instead of one group of computer hobbyists, there is a group of 8080 users, a group of 6800 users, COSMAC users, IMP-16 users, and F-8 users, etc., with no overlap of interest and no cross benefit from the development of software.

One way around this problem might be to identify the 256 most valuable and powerful instructions we might desire in an "ideal" 8 bit processor. We could then develop a firmware emulation (i.e., a microprogrammed processor) that would convert this universal instruction set into a series of subroutine calls specific to each particular chip. This alternative trades speed for universal applicability of software, memory efficiency and ease of assembly level programming. It also has the benefit of providing a common target for all chip manufacturers. Each manufacturer might be expected to microprogram his future processors to most efficiently execute this standard set of 256 instructions. . . . [The main problem is picking the right "universal instruction set."]

. . . We should recall that today's microprocessors are superior to the largest computers of 20 years ago. In 20 more years, we may have home computers *equivalent* in power to the IBM 370/168 or CDC STAR or even the Illiac IV. We should therefore think big and insist on upward compatability of any software we develop.

Michael A Sicilian
Reston VA

INSTANT ADDICTION

Last weekend my husband discovered BYTE among his brother's magazines. That did it! I didn't see his eyes the rest of the weekend because they were hidden behind your magazine. I know it must have been interesting because he read through Saturday afternoon nap time AND Sunday afternoon nap time. I'm sure you don't realize just what an unbelievable feat this is. (My only exposure to the magazine was the outside cover, which is as close as I dared get.)

Now, at my husband's insistence, I am enclosing a check in the amount of $30 for a three-year subscription to BYTE . . . but, would it be possible to have the subscription begin with the first issue of the magazine since my brother-in-law wouldn't let us have his copies of the first three issues?

Mrs J Ryon Walker
Oklahoma City OK

P.S. I'm not jealous of your magazine – just anxious to do some reading in it myself!

BYTE is very much in demand. We cannot promise you back issues, but you have been entered on our back issue waiting list. After the recent completion of a final bulk mailing of issue Number 1 to charter subscribers, we're down to about 400 copies of that issue. These will be shipped to the waiting list on a first come, first served basis. When Number 1's are out, Number 2's and Number 3's will be shipped to the waiting list in the same fashion.

INTERCHANGE STANDARD?

I am amused by your complaints about the standardization (or lack thereof) of computer interfaces. An interface standard exists which should be ideal for small, moderate speed (up to 1 megabyte/sec),

byte serial/bit parallel data transmission. It allows up to 15 devices to be connected by up to 20m (65.5 ft) of cable. That's one interface connection to provide control of all 15 devices, *not* one interface board each, as some systems require. Furthermore, the organization looks like it ought to make the design of a true IO channel facility pretty easy. (An IO channel is a piece of hardware which executes a set of commands involving the control of IO devices and the transfer of data more or less transparently to the main program. Since it takes care of the bookkeeping involved in counting bytes and updating the storage address to/from which the data is moved, only one memory access per byte is needed instead of 3 or 4; a great savings indeed.)

A copy of the standard – IEEE Standard Digital Interface for Programmable Instrumentation STD 488-1975 – can be obtained from

IEE
345 E 47th St
New York NY 10017

for $10 to non-members.

If we could get enough people to agree that they like it, I think that this system is really the way to go.

I hope your magazine doesn't run out of interesting things to say; so far, it looks great. Have fun.

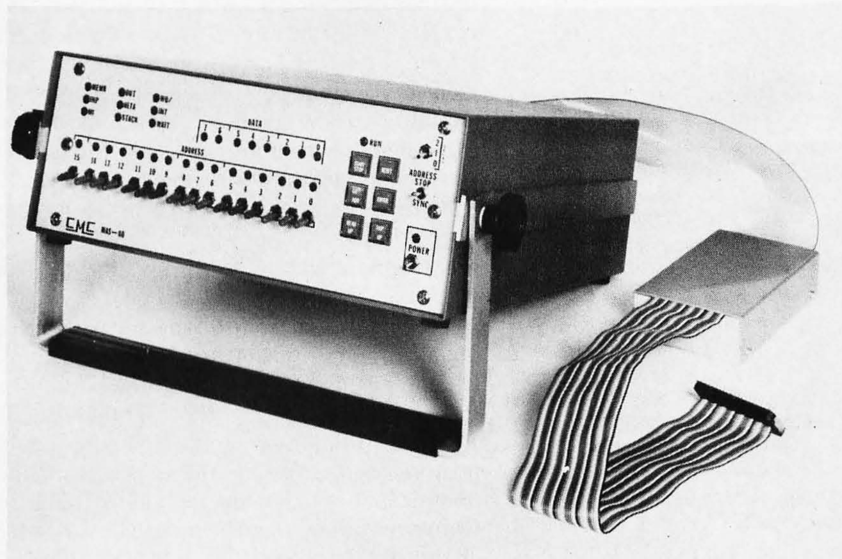
Roy Hinman
Chicago IL

REFLECTION AND ANALYSIS

I think this time of the year is prone to reflection and analysis (among a lot of other things). In the past year microprocessors and the related digital devices for the man in the street have come a long way. It is hard to pin down a good birthdate (like MITS's entry into the market, or Titus's article on the 8008, or maybe the TV Typewriter), but about a year ago most electronic hobbyists were just beginning to see the field in its own right. Now in one year we have some very good sources of information to work with, good sources of parts (and getting better), and some imaginative people in the field.

I think a major reason for the success comes from two large groups. The first is the frustrated electronics experimenter who used to either build gadgets or repair TVs or was into amateur radio. Many wouldbe radio operators stayed away from amateur radio possibly because of the cost, the FCC tests, or maybe even the FCC scares them. Many people give me just those reasons. Another

What's New?



ATTENTION HARDWARE EXPERIMENTERS, COMPUTER REPAIR TECHNICIANS

What do you do when you're not quite sure what your 8080 system is or should be doing? How can you develop an 8080 system using a full function control panel when the ultimate use will be without the control panel? One answer of course is to use a simulator which provides a CPU plus "something else." An example is this MAS-80 Microcomputer Analyzer System by California Micro Computer. The system contains its own 8080 chip which is protected against damage via buffer circuits. Before plugging the 8080 into a new system or to gain a full service front panel in an existing system, simply substitute the 40 pin dual in line output connector of this analyzer for the normal 8080 chip. The "something else" provided by this product includes single step execution logic, address matching features, control and monitoring of the 8080 signals. The unit also includes space for a limited amount of user developed custom circuitry.

This $850 analyzer could be the beginning of a new breed of test bench equipment designed for the microprocessor field. For more information, contact CMC at 9323 Warbler Av, Fountain Valley CA 92708.

group contains the persons who somewhere along the educational path have done some programming and got hooked on telling a machine what to do. There are a hell of a lot of people out there who have done some BASIC or FORTRAN coding, even if it was just reading and printing cards for Snoopy calendars. Those I've met who have interest in personal computing have great anxiety about the hardware. Computer clubs with a few hardware heavies will help some of that anxiety in the future, I hope. I think clubs are an important part of the hobby's future.

I think the best thing the computer hobby has going for it are the standard logic parts that don't require a MS in EE to use. Finally, an electronics hobbyists has a really wide open field to play with at the level that high schools can sponsor. (I guess that's a prediction – in the far future, maybe five years.)

Your editorial and Ryland's article point up an important issue – software vacuum. One point that can distinguish the hobbyists' market from the big time is the use of PROMs. I think many software houses could protect themselves a little better from the unethical hobbyist by providing plug in firmware rather than listings. The software could still be copied, but I think plug in compilers or interpreters will have a definite role in the future. Eventually, the plug in firmware would be more effective than copying the listing and running in RAM. That may not develop, but speculation is fun, anyway.

Gary Liming
Florissant MO

There are a hell of a lot of people out there who have done some BASIC or FORTRAN coding, even if it was just reading and printing cards for Snoopy calendars.

ENTHUSIASM

On the overwhelming recommendation of two close friends and perusal of the first four issues of BYTE, here is $12 for a subscription. So far I have learned more from the articles in BYTE than I have been able to learn from any other source. In addition I have been able to decide what to do to get my feet wet in the microprocessor area. You have a very professional format for the magazine and very well edited articles. In particular you do not fall into the technical magazine trap of editing out the authors' enthusiasm, something that went out of technical writing about 50 years ago.

I look forward to long association with the magazine as a subscriber and perhaps a contributor.

Bill Harding
State College PA

EXPLODING CIRCUITS

They say that experience is the best teacher and that practice makes perfect. In a couple of microseconds my computer taught me the worth of both these statements. I had my computer up and running when I decided to wire in another circuit. It seemed like it would be a simple task since both the computer and the other circuit were working. The only problem that could develop I figured I could eliminate by connecting an extra ground line so that I would not develop any ground loops. I had just made the connection to the computer when the other end of the wire fell against the 110 VAC where it comes into the computer. Needless to say I had left the computer plugged.

I never knew that integrated circuits could explode.

The power switch was off but you should have seen the arcing that happened between the closely spaced foil paths on the printed circuit board. By the time I pulled the plug there was smoke pouring out of my computer. I never knew that integrated circuits could explode. What happened was that I delivered 110 VAC to every ground pin in the computer. Some of the integrated circuits that were further from the point of 110 VAC entrance did not suffer physically but there are a few that do not seem to work the way they should. I know my 1101 memory circuits are doing some funny things.

On my next computer I am going to do things differently. All the high voltage sources are going to be physically isolated from the computer and they will be plainly marked. By high voltage I mean anything that the computer doesn't use directly, such as 110 VAC or even 10 VAC. The second thing that I am going to do is install some outlets on my bench that I can turn off easily with a switch. That way I can leave my test equipment running and when I need to go into a piece of equipment all I need to do is throw the switch removing all power to the circuit. I might even install an audio alarm telling me when the power is off. You see, I have a terrible memory and if there is some way that I get feedback telling me it is unsafe to go into some equipment it might stop me from doing something stupid.

I hope my experience has taught me something. Maybe two attempts at computer

building isn't enough practice to become perfect but I will do a better job the second time. With all the new circuits coming out, my first computer had become something of an antique; so its untimely passing may serve more of a purpose than if it had lived on.

[Name withheld due to embarrassment]

LIGHT PIPES FOR A LIGHT PEN?

I greatly enjoyed your article on light pens; it will be a useful addition to any system. However, a few possible modifications could eliminate some of the problem areas you mentioned. How about using a fiber optic light pipe mounted in the pen? This would eliminate any loading caused by a long wire run to the control circuitry and it would narrow the area detected by the photo cell without lenses. The sensitive area would be only a small fraction larger than the diameter of the light pipe end. A plastic type of fiber would be better than glass, due to its greater flexibility and lower cost.

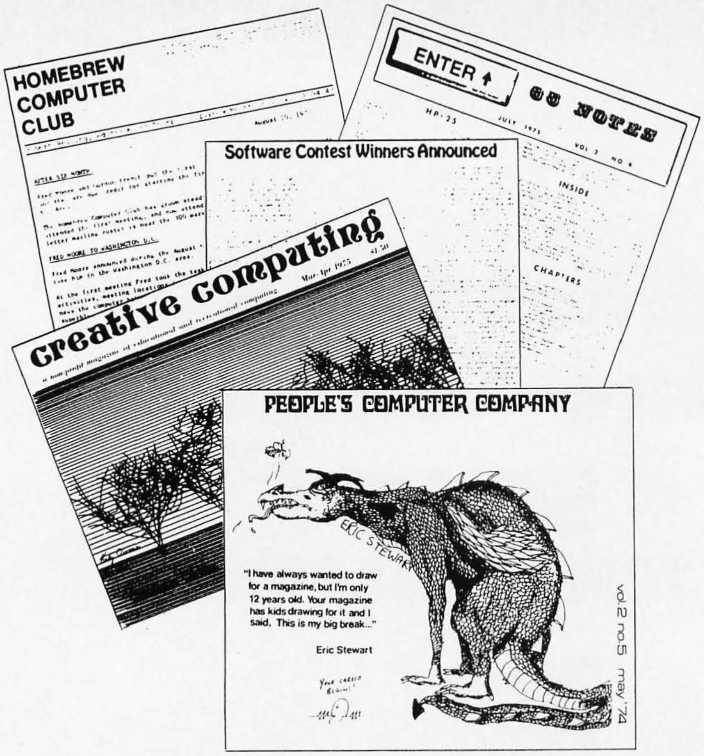
David Rawson
1825 Gary
Wichita KS 67219

PS: Anybody in my area working with minis, please get in touch.

MICROPROGRAMMED PROCESSORS, ANYONE?

I really dig the fact that you have a magazine dedicated to the novice; I can be learning software while I am working on my hardware. Enclosed you will find my application for subscription. One thing I would like to see is articles about "do-it-yourself" microprocessor systems, such as the Intel 3000 series, which uses a control unit and an array of central processing elements to implement the microprocessor. The main advantage to this system, as I see it, is that you can build a system as powerful as you like (or as weak as you like); and you can configure it to match any other microprocessor. The main disadvantage, though, is that you have to program your own microinstructions, but that might be a blessing in disguise, because you can select the best of the instructions from the other processors, and use them in your own. But that's enough from me — I'd like to see what other people have to say about it.

Steve Allen
Portland OR



Clubs and Newsletters

New Brochure from HP-65 Users Club

The HP-65 Users Club is a non-profit volunteer group of enthusiastic programmable calculator users, designed to provide a source of information for the owners of such machines. The club has just printed a new brochure describing its objectives and activities. So far the club has centered on the HP-65; but programs and techniques concerning the HP-55, HP-25 and other programmable calculators have appeared in the newsletter **65 Notes** which is published monthly. To write for the brochure (or subscribe to **65 Notes** at $12 per annum) write:

HP-65 Users Club
Richard Nelson
2541 W Camden Pl
Santa Ana CA 92704

Project SOLO

An experiment in applications of computers to education, Project SOLO is located at the University of Pittsburgh. The curious should write to Soloworks, Thomas A Dwyer, Professor of Computer Science, University of Pittsburgh, Pittsburgh PA 15260.

Hudson Regional Computer Fair

Computer Fairs? It just had to happen – the beginning of computer fairs for high school students, analogous to science fairs. BYTE received an announcement of the Hudson Regional Computer Fair, March 27 1976 at the White Plains Public Library, White Plains NY. Computer fairs are intended to promote and encourage the imaginative use of computers by elementary, middle and high school students. The White Plains event is sponsored by the Library, Comput-O-Mat Systems and Wang Laboratories. Entry blanks and guidelines can be obtained at the Reference Desk of the White Plains Public Library, 100 Martine Av, White Plains NY.

A Club In Michigan?

C J Lamesfield (Box 271, Davison MI 48423) would like to contact individuals interested in forming a computer club in his locale.

New Jersey's Amateur Computer Group

The **ACNJ NEWS** is the information organ of the Amateur Computer Group of New Jersey. The December issue reports that the November meeting of the group included demonstration of a small 6800 system by Jim Loy of Motorola. A geographical breakdown of members by county and state was also published. A summary of member systems showed that of the 129 members, 52 have microprocessor systems at home. Other activities of the Garden State organization include a software library run by Tom Kirk and amateur computer courses arranged through the Union County Technical Institute. Group purchases are also being arranged with Carl Reisinger coordinating the activity. For information on future happenings, contact Sol Libes at (201) 889-2000 (x-248, x-247 or x-282) days, and (201) 277-2063 evenings. Meetings are held the third Friday of every month – even months at Union County Technical Institute, odd months at the Middlesex County College.

Seattle Club Activity

Bob Wallace, PO Box 5415, Seattle WA 98105, reports that a club is forming in the Seattle area. For information, call (206) 524-6359 11 AM to 5 PM.

British Amateur Electronics Club

One organization which has been going strong on electronics themes for some time is the British Amateur Electronics Club. The **BAEC Newsletter** will be of local interest to those who live in the British Isles, and to the English speaking people in Continental Europe. (The fact that the newsletter is oriented to the English language will make it of interest to Americans in BYTE's audience, people whose second language is more apt to be Fortran, Basic, Cobol or PL/1 than a people to people language.) According to the flier sent to BYTE, the **BAEC Newsletter** has published news of electronics and ideas for projects since 1966. A yearly exhibition is held by the organization, and local chapters are found throughout Britain. Recently, major projects have been made a part of the quarterly newsletter. The first such project was the BAC Naughts and Crosses Computer. The BAEC Computer is the current major project; the goal is to have a general purpose computer design at the conclusion. Fees for yearly BAEC membership depend upon where you live:

£2.00 Sterling for one year, UK only.
£3.00 Sterling for one year, foreign, surface mail.
£4.00 Sterling for one year, foreign, air mail.

Write to:

The Hon Secretary, BAEC
Mr J G Margetts
11 Hazelbury Dr
Warmly, Nr Bristol
E N G L A N D

San Diego's Newsletter

Personal Systems is the newsletter of the San Diego Computing Society. Issue 4 of volume 1 was received at BYTE recently and includes information on the San Diego society's group purchases, an essay on the history of the field to date by Lance A Leventhal, a book review and other information. The address for inquiries is:

Personal Systems
10137 Caminito Jovial
San Diego CA 92126

Milwaukee Hobbyists?

Will Piette, W266 North 7060 White Oak, Sussex WI 53089, is looking for compatriots in the Milwaukee area. Anyone interested in starting a Milwaukee area club should contact Will.



The UCLA Computer Club

Michael S Maiten, 3135 Barry Av, Los Angeles CA 90066, sent in a note about the UCLA computer club. This is a campus-only club, which allows students and staff at UCLA to get time on the Campus Computing Network. According to Michael, the club is mostly software oriented (IBM 360/91, DEC machines). The club owns its own DEC classic LINC computer and may get more hardware in the future. The club offers free classes in the evening; however, membership is limited to UCLA students, faculty and staff only. Michael is the current secretary of the club. The club also has its own IBM card form for use with traditional computer center input output techniques.

Want a Computers-Only Classified Advertising Publication?

D H Beetle, 24695 Santa Cruz Hwy, Los Gatos CA 95030 is in the process of initiating a classified advertising specialty publication called **ON LINE**. The idea is to attract individuals' advertisements of personal surplus equipment with low cost printing and an inexpensive subscription rate of $1 for four issues. (18 issues, 1 year cost $3.75). The result should be much like one of the neighborhood classified advertising publications which occur in many regions of the country. The only difference is that the "neighborhood" in this case is technological, not geographical.

Association for Educational Data Systems

The Association for Educational Data Systems is an organization concerned with computer uses in education. The AEDS publishes a journal called **AEDS Monitor** on a quarterly schedule; the orientation is towards secondary school and college educators who utilize computers in their courses. AEDS is located at 1201 16th St NW, Washington DC 20036.

Is your organization listed here? Do you want to start a local club? BYTE wants to encourage the transfer of information to and among the practitioners of the personal information systems art. If your club or organization is not mentioned here, be sure to put a member in charge of sending us the information on your activities; if you're interested in starting a club, tell us and we'll help out by printing your name and address.

New England Computer Society

The second meeting of the New England Computer Society was held December 3 1975 at 8 PM in the cafeteria of the Mitre Corporation, Bedford MA. The steering committee reported on activities concerned with formal organization. A constitution based upon an American Radio Relay League radio club model was presented for discussion.

As at the first meeting, the club auctioned off copies of Intel documentation supplied by Gary Carrol of Intel in order to fund printing and mailing a newsletter to the 200 or so persons on the local mailing list.

The technical presentation was provided by Don Rosenthal and Chris Ryland in a short talk describing prospects for the "HARPA" network: Ham's ASCII Radio Protocol Agreement. The idea is to begin development of an amateur radio version of the ARPA network so that microcomputers across the country could be linked by radio into an automated packet switching network. One purpose of the talk was to solicit interest from active hams in the society. A show of hands revealed that roughly 50% of those present had ham tickets, and several were recruited into the project. One input to the project came from several amateurs who were experienced in the field of automated two-meter repeaters, which could be used as a starting point for the design of the microcomputer radio networking technology.

The next meeting of the NECS was scheduled for January 7 at the same time and place. Until a permanent mailing address is established for NECS, inquiries should be directed to Carl Helmers at BYTE Magazine, Peterborough NH 03458.

PCC's Special Games Issue

Word from Peoples Computer Company is that the January 1976 PCC is a special issue devoted to games. It includes BASIC versions of three classics on a theme of space exploration and adventure:

- Star Trek for an HP2000, with a complete listing on four pages, plus descriptions and sample runs.
- MOTI – A "save the empire from the alien MOTI invasion" game.
- RESCUE – A game of suspense in which the player guides the rescue of stranded astronauts.

Find out the latest information on Tiny BASIC and other PCC projects by latching onto this special 40 page issue. Single copies are $1. PCC publishes a newspaper style publication several times a year, and is located at Box 310, Menlo Park CA 94025.

North Texas Hobbyists

The Computer Hobbyist Group of North Texas puts out a newsletter edited by Bill Fuller and Neil Ferguson. The club meets regularly and has had some interesting speakers. Contact Bill Fuller, 2377 Dalworth No. 157, Grand Prairie TX 75050.

N	F	S		B	R			T		N		S			P
	R		C		R	D		M		N	S			N	R
T			T	P		T	R		T		R		P		
W	R		T		X	T		R	N		L	R			G
	D		C		D		C	N					N	S	R
	L	B			R		V	S	G		B	W	C		
C				D	N		P		N	T	R		H	C	M
N	T	N	M	R	T		C					N			H
	C	T		P	C			C	M	G	R	D	N	R	T
L			T		L	S	R				Y	N	T	P	
	M	G			S		N	L	L	T	N		R	P	R
V	P		D		D	P	C	L			L	F	Y		
	L	R	R	L					S	L			R	R	G
		P		T	Y	P			T		N		B	R	
Q	X		S	C		L		R	L		T		R		L
	F		N	C	T			N	G		S	S		Y	L

Space Ace

By inserting the missing vowels (a, e, i, o, and u) in the appropriate blanks, all 50 words from the list will fit into the matrix. As you find each word and insert the correct vowels, circle the word in the matrix and cross the word off the list. Words may be forward, backwards, up, down, or diagonal, but always in a straight line, never skipping letters. However, some of the letters are used more than once. After circling all the words on the list, the seven remaining letters (including two blanks for vowels) in the matrix will spell the name of a high level computer programming language these words are related to. Be careful, though; some words may appear to fit in more than one place in the matrix. There is, however, only one correct position for each word, so that all the words from the list will be used.

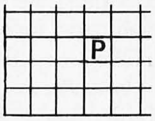
Robert Baker
34 White Pine Dr
Littleton MA 01460

ANSWERS TO FEBRUARY NUMBERS page 69

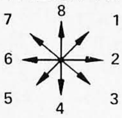
Here is the answer to Robert Baker's NUMBERS puzzle in the February BYTE: The 12 letters left over spell "GROUND HOG DAY." In case you had troubles finding the words (or you like to cheat), the following list will indicate where the first letter of each word is located in the matrix and the direction the letters go from the first letter. The location is given as an X,Y co-ordinate in the matrix, with the letter P used as the reference point (X=4, Y=3). The direction code indicates the direction the letters go from the first letter as follows:

(X, Y) CO-ORDINATES

4, 3



DIRECTION



BILLION	13, 2	7
BINARY	1, 4	2
COMPLEX	7, 10	3
DECIMAL	3, 2	2
EIGHT	13, 9	4
ELEVEN	10, 15	2
EXPONENT	2, 3	2
FIFTEEN	3, 11	4
FIFTY	3, 9	8
FIVE	12, 11	6
FORTY	15, 1	7
FOUR	15, 1	8
FRACTION	1, 6	8
HUNDRED	15, 8	8
IMAGINARY	14, 12	5
INTEGER	6, 1	2
LOGARITHM	14, 13	6
MILLION	2, 15	4
MODULUS	6, 13	3
NEGATIVE	3, 15	2
NINE	5, 6	5
ONE	2, 5	8
POSITIVE	6, 14	2
RADIX	5, 14	6
RATIONAL	5, 14	4
REAL	15, 4	8
ROOT	14, 1	8
SEVEN	5, 1	6
SIX	1, 5	3
TEN	2, 8	3
THIRTEEN	6, 5	8
THIRTY	5, 5	1
THOUSAND	14, 4	8
THREE	7, 6	3
TRILLION	4, 13	4
TWELVE	8, 4	1
TWENTY	13, 12	6
TWO	9, 9	7
ZERO	13, 1	7

What's in a Video Display Terminal?

Don R. Walters
3505 Edgewood Dr
Ann Arbor MI 48104

Let's look at the video display terminal as a black box which is connected to a computer system (somehow) as depicted in figure 1. Since the computer system has already been explained (at the block diagram level) in BYTE ("The State of the Art" by Carl Helmers, November 1975, page 6), we will concentrate on what smaller black boxes make up a video display terminal.

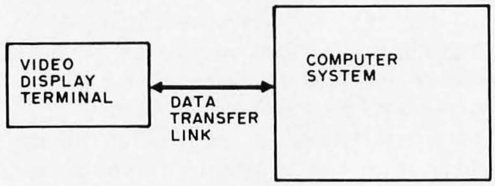


Figure 1: Two black boxes: the video display terminal and the computer system.

Figures 2 and 3 illustrate subassemblies typically combined to form the video display terminal. We see that the video display terminal is actually made up of some more familiar subassemblies, such as a keyboard, video display controller, video display, and a parallel (figure 2) or a serial (figure 3) interface. Let's take a closer look at each subassembly and see what its function is. The keyboard is a man-machine interface which is used to enter data (alphabetic commands, instructions, and/or numbers) into the computer system. When a key is pressed, the equivalent electrical code assigned to the character is generated and is available in parallel form. Thus all bits of the character's code are available at the same time at the output of the keyboard.

Now that the electrical codes of characters can be easily generated using the keyboard, how will that data be transferred to a computer system? Since the data from the keyboard is already in a parallel form, the data could be transferred to the computer system through the parallel interface in figure 2. The parallel interface handles the buffering of the data between the keyboard and the computer system (which must also have a parallel IO interface). The parallel data from the keyboard could also be sent to a computer system in serial form by using the serial interface of figure 3. Serial interfaces are usually used when the data path between the video display terminal and the computer system is longer than five feet, which would be the case when the video display terminal is to be connected to an acoustical coupler. The coupler is a device which changes serial data into frequency shifted tones to transmit the data over voice grade telephone lines so that a terminal can be used with a remote

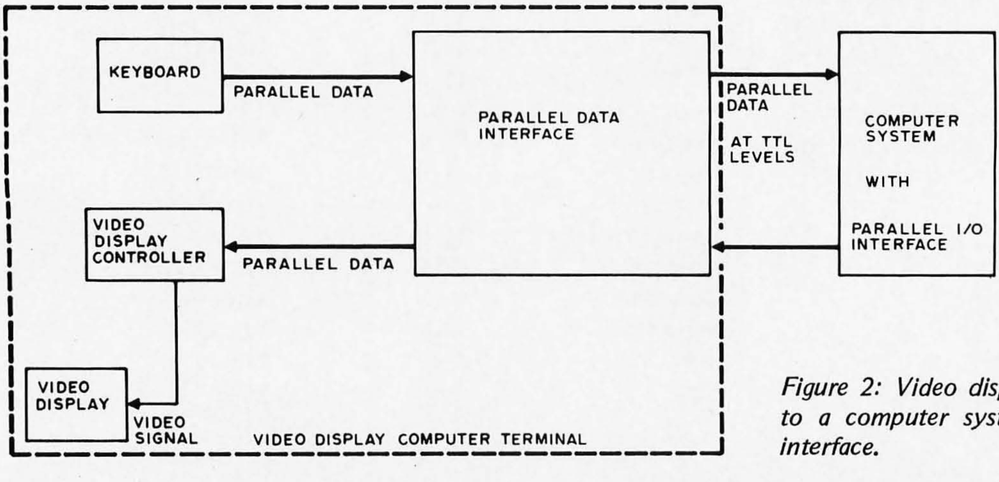


Figure 2: Video display terminal interfaced to a computer system through a parallel interface.

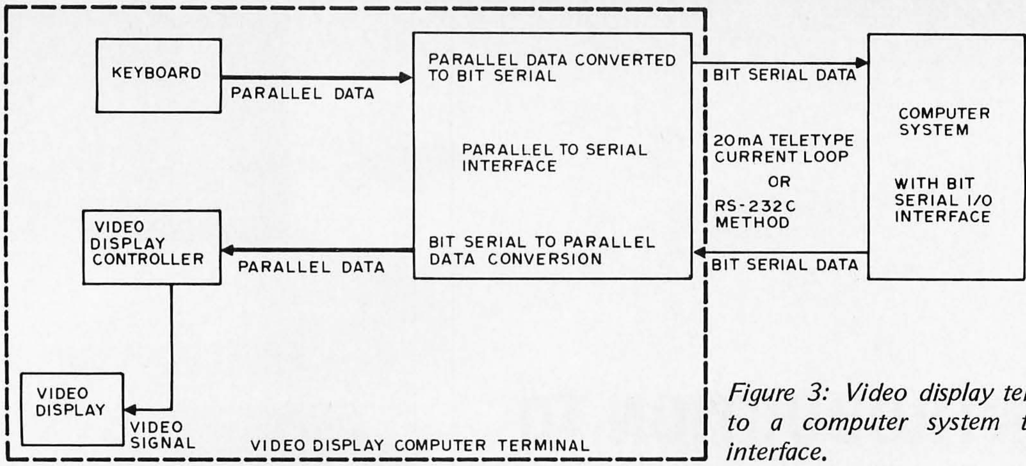


Figure 3: Video display terminal interfaced to a computer system through a serial interface.

computer system via the telephone. The serial interface converts the parallel data from the keyboard to a bit serial form. In this form, the bits of the character are sent one bit at a time until the entire character has been sent. Of course the computer system must also have a serial IO interface.

We now have traced the data path from the keyboard of the video display terminal to the computer system. Let's trace the data path from the computer system to the video display terminal. Data is sent from the computer system to the video display terminal in parallel or serial form with the same type of interface (parallel or serial) as is used between the keyboard and the computer system, except that each data path must have its own interface electronics.

The data from the interface (parallel or serial) is fed to the video display controller in parallel form. The video display controller converts its parallel data input to a composite video signal which causes the video display to show the desired characters.

The video display portion of the terminal is essentially a TV set without the RF tuner, IF amplifiers, and mixer circuits, but with the necessary circuitry to display a video signal on a CRT screen.

As you can see, the video display terminal is not a very complicated black box after all.■

John M Schulein
Homebrew Computer Club
P O Box 626
Mountain View CA 94042

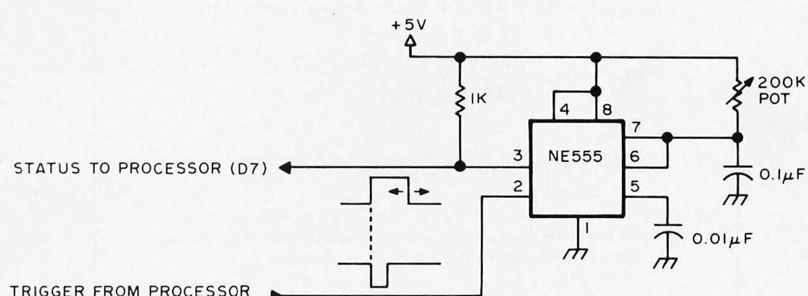
Pot Position Digitizing Idea

A scheme to convert the position of a potentiometer arm into a digital value, using a cheap commonly available timer IC (NE555) and a few bytes of program in an 8008 or 8080 microprocessor, is shown in figure 1. The software is organized as a subroutine and uses the flags and the A and B registers. The NE555 is triggered by the OUT TRIGGER instruction and then the program monitors the output pin of the NE555 in a loop that increments the B register. When the NE555 times out, the program exits from the subroutine and the B register contains a digital representation of the pot position.

The hardware and software shown in figure 1 was run on an 8008 system with a 2.5 μs clock and the B register digital output varied from 2 to 65 Hex. The values of the pot and/or the timing capacitor can be modified (see the NE555 data sheet) to suit your processor's speed and the desired range of the digitized output.■

Figure 1: Pot Position Digitizing Idea.

Hardware



Software

```
POTPOS:  MVI   B,0
         OUT   TRIGGER
CONT:    INR   B
         IN    STATUS
         ANA   A           ;Sets sign flag
         JM    CONT
         RET
```

NOTES:
1. Software written as a subroutine for the 8008 or 8080 microprocessors.
2. The flags and registers A and B are affected by the subroutine.
3. Register B contains the pot position on exit.

What's New?



PRINTERS, ANYONE?

One solution to the hard copy problem is to use a printer such as this General Electric TermiNet 30 matrix teleprinter. This type of equipment is normally leased by firms employing computers as a time sharing link; however, the standard RS-232 control interface could be used to link to a microcomputer instead of the normal modem. If you're interested, the cost of a one year lease is $88 per month for an 80 column KSR version. Contact GE Data Communication Products Department, Waynesboro VA 22980, (703)942-8161.

TERMS: Add 50¢ to orders under $10. Cal + tax. Sorry, no COD.

GODBOUT
BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

BANKAMERICARD® MASTERCHARGE®: call (415) 357-7007

HOBBYWRAP TOOL $41.95

Why solder your prototypes, when there's a better way? Wrapped connections protect components from heat, are easily modified for corrections or changes, make connections that are better than solder, and can speed up the time required for the assembly of complex digital projects. If the high cost of wrapping has put you off, take a look at our offer...you get the tool (rechargeable so you don't have to deal with trailing cords in tight places), bit, charger, nicads, and instructions.

FOR PRICES ON OTHER PARTS (LIKE 2111s & 2112s) SEND FOR OUR FLYER

INDIVIDUAL PARTS... AT QUANTITY PRICES

8008 Microprocessor---first of the 8 bit CPUs...........$17.95
8080 Microprocessor---So far, the most popular 8 bit CPU..$29.95
5204 4K EROM---Static, non-volatile, erases with UV light..$17.50
5204 pre-programmed with 8080 monitor routine...........$29.95
(note: if you'd like one of these in the 8080 chip set below---instead of an unprogrammed 5204---add $10 to the chip set cost)

10,000 uf at 10 volts

$1.25 --- The perfect tonic for power supplies with a farad deficiency.

LOW POWER SCHOTTKY

74LS00	Quad 2 input NAND gate	.40
74LS04	Hex inverter	.45
74LS08	Quad 2 input AND gate	.40
74LS10	Triple 3 in NAND gate	.40
74LS20	Dual 4 input NAND gate	.40
74LS42	BCD to decimal decode	1.25
74LS138	1 of 8 decoder	1.40
74LS168	Dec syn up/down cntr	1.50
74LS169	Bin syn up/down cntr	1.50
74LS175	Quad latch	1.38

CHIP SETS

all 2102s less than 750 ns!

8008 CHIP SET $32.50

(1) 8008 CPU (8) 2102s
Note: at this low price we cannot include data.

8080 CHIP SET $59.95

(1) 8080 CPU (8) 2102s (1) 5204 ROM
Note: at this low price we cannot include data.

PACE CHIP SET $125

(1) PACE CPU (4) DS3608 (1) DS0026
(1) DM8837 (2) 5204 ROMs
+ PACE DATA PACKET

DELUXE PACE SET $195

(1) PACE CPU (4) DS3608 (1) DS0026
(1) DM8837 (32) 2102s (4) 5204 ROMs
+ PACE DATA PACKET

OUR PRICES GIVE YOU A PLEASANT JOLT:

POWER CARD $44.95

Not just for powering JOLT--- but for 8080s, 6800s, 8008s, etc. Enough juice to power the JOLT CPU plus 16 Kbytes of memory (like four of our NAKED RAM boards). Includes crowbar overvoltage protection on +5V, heat sink, and transformer. Same size as JOLT supply card.

NAKED RAM $79.95

This general purpose 4K x 8 board is designed specifically for JOLT systems, but is equally applicable to other bi-directional buss systems. If you don't need the onboard regulation or buffers of our "bigger brother" 4K x 8 board, then this is the way to implement cost-effective memory. Low power --- under 750 ma. Same size as JOLT card.

BOTH KITS INCLUDE INSTRUCTIONS

CAVE grafix™

Altair 8800 Plug-In Compatible Memory Kits

4K x 8 RAM Board

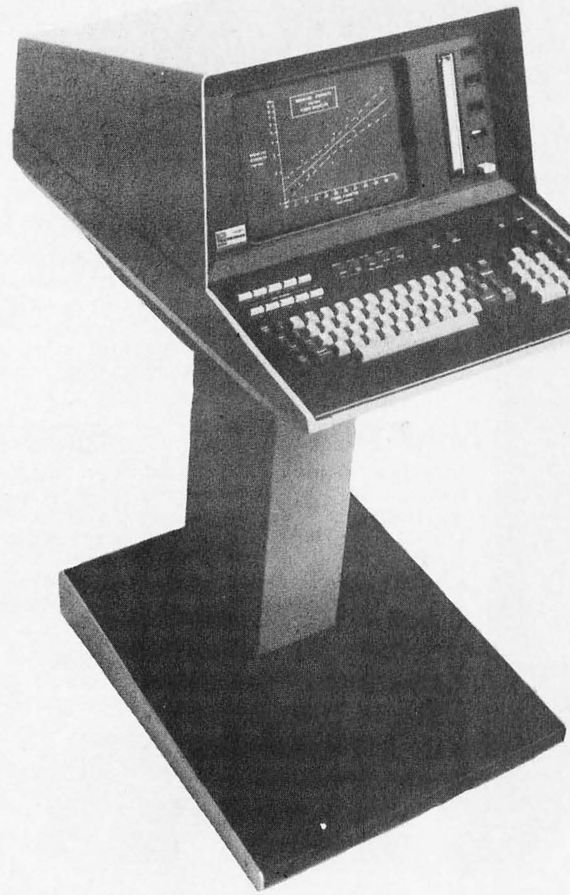
$109.22 --- A THIRD OF A CENT PER BIT. Our most popular memory kit includes sockets for all ICs, lots of bypassing, low power consumption (UNDER 1A!) with 5 regulators to share the load for minimal heat generation, quality plated-through board. Buffered addresses present 1 LPTTL load; outputs drive 20 standard TTL loads. Typical 500 ns. access time at 25°C. With assembly instructions and logic print.

EROM Boards

NOW YOU CAN STUFF UP TO 8K OF SOFTWARE---EDITOR/ASSEMBLER OR WHAT HAVE YOU---ON A BOARD. Our EROM boards feature the same qualities as our RAM board: sockets, lots of bypassing, buffered addresses and outputs, on card regulation, and more. Our EROM boards are expandable, too; you can start with a 2K board, and populate it with more EROMs over a period of time until you've got the full 8K. PRICES: 2K by 8---$115; 4K by 8---$159.95; 8K by 8---$249.95. All EROM boards feature low power consumption: the full 8K requires only ½A at 5 volts & 150 ma at -12 volts. With instructions and logic print.

8080 Software Board

So you've got yourself a microcomputer...only to find out you need software to run it efficiently. Give your 8080 editor, assembler, and monitor routines with our 8080 software board. It uses the same configuration as our 4K by 8 EROM board, but the EROMs are pre-programmed and dedicated to making happier 8080s. Extra feature: because this board is EROM rather than mask based, you can make changes if desired...so we include 2 free updates in the event of corrections or patches. The cost is $159.95. Kit includes instructions and software package.



What's New?

8080 CROSS ASSEMBLERS, ANYONE?

● FBE Research Company, Box 68234, Seattle WA 98168 sent along a press release describing the CAL-80 cross assembler which runs on a DEC PDP-8 minicomputer to assemble statements for an Intel 8080 computer. The output is a symbolic program listing and an Intel "BNPF" format PROM programming tape. A complete package of listings and manuals is quoted as $100 postpaid.

● An assembler has been written to provide nearly all the features of the MAC80 assembler for those having access to an IBM 360/370 installation. This assembler is written in FORTRAN/BAL and executes in under 100K in a VS/OS environment. Extra features include direct punching of BNPF decks for PROM programming, object code listing in both octal and hex, and ASCII constant generation with the parity bit either set or reset. The MASM80 assembler provides full MACRO and conditional assembly capability. A source deck and documentation is available for $55. Contact Well Test Data Inc, PO Box 7081, Kansas City MO 64113.

TEKTRONIX = BASIC + GRAPHICS;

Here is an example of a total package of computing power designed for individualized use. The product is a Tektronix 4051 BASIC Graphic Computing System. For a purchase price of $6995 (also available on a lease plan) you will receive a complete computing system which features a BASIC interpreter, mass storage and alphanumeric/graphic display with enough resolution to make an excellent space war display. Specifications of the standard machine include:

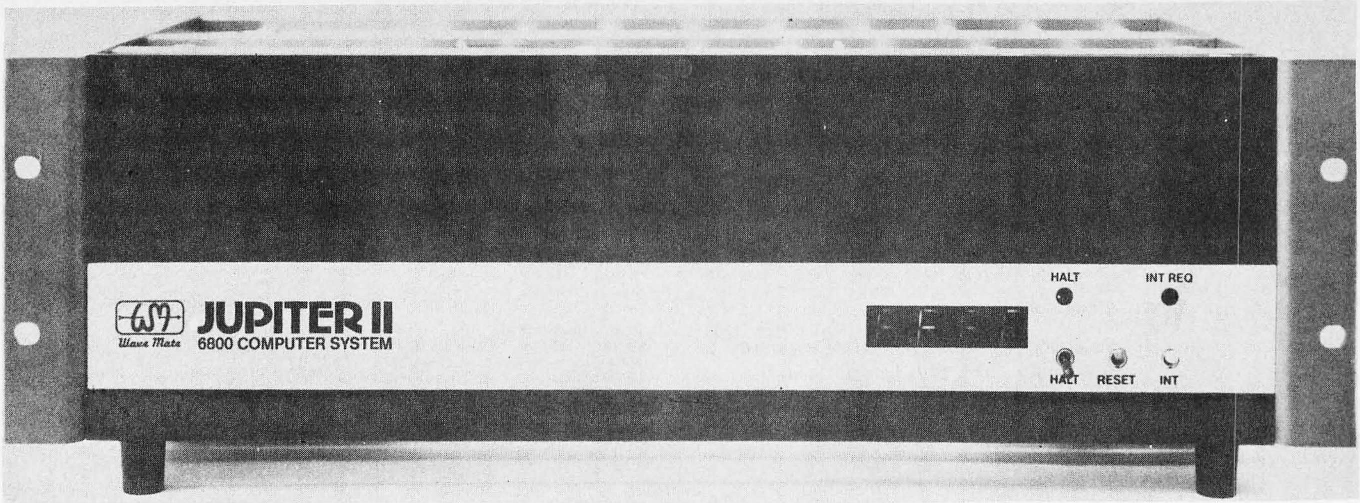
- Graphic resolution of 1024 by 780 points.
- Alphanumeric display of 35 lines by 72 characters, upper and lower case.
- BASIC workspace of 8 KB, expandable in increments of 8 KB to a maximum of 32 KB.
- Ten user definable key functions with shaft capability for 20 programmable functions.
- Built in 3M cartridge tape drive which can store up to 300K bytes per cartridge, perfect for off line storage of programs.

The most important point about this package is that it is complete in one unit which sits on a desk top as your personal BASIC servant. Turn on the power and you have a machine capable of many functions, ranging from business data processing (possibly requiring a printer available as an option) to engineering calculations to fun and games.

Adding to its potential as the graphics component of a space war application is its optional 4952 Option 2 Joystick input for control of the game pieces. Of course space war is not the only game possible on such an excellent graphic toy, so if you can afford it this graphics computer would be an excellent investment in fun.

This machine will prove most useful, however, to the professionals and businessmen in BYTE's readership. Using BASIC as the programming language, this package can be adapted to nearly any business or professional use simply by writing programs, or adapting existing software packages written in BASIC. Tektronix supplies several software packages for the machine to help specialists use this machine. These include an advanced graphics package, statistics packages (3 volumes), mathematics packages (2 volumes) and an electrical engineering package. Others are under development.

For demonstrations of this machine, contact your nearest Tektronix field office; Tektronix Inc is located at PO Box 500, Beaverton OR 97077, phone (503)644-0161.



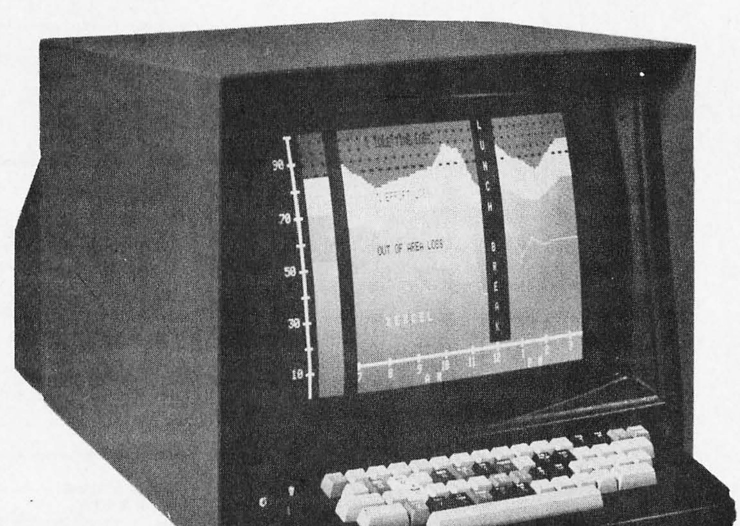
What's New?

Wave Mate sent BYTE an announcement and this picture of its entry into the microcomputer system market, the Jupiter II system. Priced below $1000, the new expandable and upward compatible system will be aimed initially at the serious hobbyist market.

Since Jupiter II is complete (ordered in kit form or assembled) the hobbyist can add simple peripherals and use the system for accounting/bookkeeping, scientific/mathematical computations, numeric control, process control, educational tool, data acquisition, inventory control, graphics, intelligent terminal or even games and music synthesis.

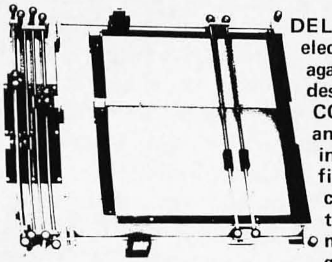
Features include: easy to test small pluggable cards, wire wrap to minimize engineering, easy bus to interface, modular plug-in power supply, all ICs socketed for easy testing and replacement, IO programming, file management capabilities and Motorola 6800 base.

The Jupiter II computer system is available from the factory 60 days ARO. For complete information contact Dennis Brown at Wave Mate, 1015 W 190th St, Gardena CA 90248, (213)329-8941.



Color video is the watch word for the 1976 computer innovation season. An excellent contribution to the field is the new Intecolor 8001 system marketed by Intelligent Systems Corporation, 2405 Pine Forest Dr, Norcross GA 30071. For $1395 you can acquire the Intecolor 8001K kit version (assembled OEM versions are also available). The result of assembling this product is an 8080 computer with 4096 bytes of user memory, PROM systems software, RS-232 serial interface, keyboard and — the most important feature — a 19 inch shadow mask color television display with character generation hardware for 25 lines of 80 colored characters. The design also includes as a self-maintenance feature the Intecolor 9-sector convergence system. It looks as if this would make an excellent starter system, coming in an integrated desk top package.

Bolt, Baranek & Newman DATA CODER



DELTA specializes in the unusual in electronics surplus, and we have once again found a very unusual, highly desirable item. This is a DATA CODER made by Bolt, Baranek, and Newman. It was made for use in the Medical Electronics field, but finds use in many different applications. It is used to digitize any type of data. The data (charts, maps, waveforms, drawings, game grids, etc.) is placed in the bed, and the sight is moved along the data. This generates Vertical & Horizontal displacement codes, which can be stored and processed by your calculator or micro-computer.
The bed size is 12¾" x 10 5/8", and has a 7" x 10½" portion which is translucent, for use as a light table. Each axis is divided into 128 increments, for a total of 16,384 discrete bits over the bed. Horiz. increment size = .10", Vert. increment size = .083". Each axis generates an 8 bit code by means of wipers on an encoding board. This makes it very easy to interface to any logic family.
This item is a must for your micro-computer, as it is now possible to digitize ANY type of data. Limited quantity. Overall size is 18¾" x 17¼" x 3" high. Shipping weight 10 lbs. With coding data.
STOCK NO. B5352 $79.95 each

HEAVY DUTY POWER SUPPLIES

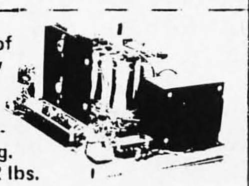


These heavy duty power supplies are ideal as a general purpose lab supply, micro-computer supply, etc. All have constant voltage transformers and large 18,000 mfd 35v. elec. filter caps for filtering. In addition, the + and – 10v and –3.6v outputs have seperate semiconductor regulators, which are voltage adjustable. Each is 22¼" long, 6½" wide, 6" high. Shipping weight 26 lbs.

B5313....Outputs at –30v @ 1.5 amp, +30v @ .5 amp, –10v @ 1.5 amp, +10v @ .5 amp, and –3.6v @ 1.5 amp. 85 watts.
STOCK NO. B5313 $24.50 each

B5314....Outputs at –30v @ 1 amp, +30v @ .5 amp, –10v @ 1 amp, and +10v @ .5 amp. 60 watts total.
STOCK NO. B5314 $19.50 each

B5342....Outputs at –30v @ 1.5 amp, +30v @ .5 amp, –10v @ 1.5 amp, and +10v @ .5 amp. 80 watts total.
STOCK NO. B5342 $22.50 each



This power supply is surplus ... a maker of computer peripherals. ... uts of +5v @ 5 amps, –5v @ 5... 5 amps, and –12v @ 20... are well filtered ... for your micro- or mini- co... are tested & working. 10½" x 13... high, shipping weight 22 lbs.
STOCK NO. B5312 $34.95

SOLD OUT

ALCO KNOBS FOR EVERY APPLICATION



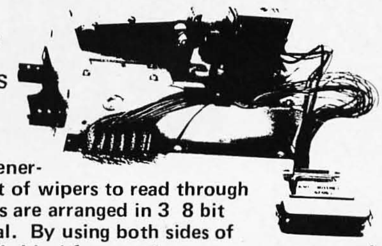
Machined aluminum knobs with 2 setscrews. Clear anodized finish. Latest distributor price $1.65 to 1.05 each.

ALCO NO	O.D. x HT.	DELTA NO.	PRICE
KS 500A	.500" x .625" w/skirt	B6176	.40 ea, 10/3.50
K 700A	.750" x .620"	B6177	.40 ea, 10/3.50
K 1250A	1.250" x .750"	B6178	.50 ea, 10/4.50

2502 AY-5-1013 UART

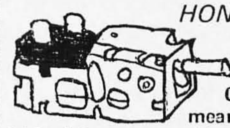
Brand new, tested, General Instrument AY-5-1013 UART (Universal Asynchronous Receiver/Transmitter). This 40 pin IC is pin compatable with Signetics 2536, TI's TMS6012, SMC COM2502, etc. The UART is the standard building block for parallel to serial and serial to parallel conversion, serial communication, etc. A must for TVTs, cassette interface, RS-232 interface, etc.
STOCK NO. B3119 $3.95 each

SECURITY SYSTEM CARD READER



Another unusual item from DELTA! These card readers were made by AMP, Inc. for use in SECURITY SYSTEMS A stiff card 2 1/8" x 3 3/8" (credit card size) is inserted, which closes a micro-switch. A 115v AC solenoid is then energized, which pulls down a set of wipers to read through holes in the card. The wipers are arranged in 3 8 bit bytes + 1 bit, for 25 bits total. By using both sides of the card, 48 bits are possible! Ideal for security systems---entry can be controlled by use of a card, with an almost infinite number of combinations, rather than an easily duplicated key. As another example, an entire Social Security no., plus an entry code, could be read from a card. Many more applications are possible. 5" x 5" x 9" deep. 6 lbs.
STOCK NO. B5353 $19.95 each, 2 for $35

HONEYWELL HUMIDITY CONTROLLER

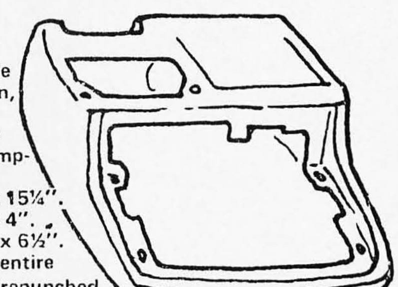


Made by HONEYWELL for automatic control of humidifiers or de-humidifiers. Control range 10% RH to 60% RH, sensing by means of a nylon ribbon. Snap action switch with 240v AC contacts. Ideal for home lab, or greenhouse. 3½ x 2¼ x 1¼.
STOCK NO. B6263 1 lb. $4.95 each, 3/14.00

650 WATT POWER TRANSFORMER

This versatile transformer has dual primaries, for 115v or 230v operation, or for isolation use. Each is also tapped for low or high line. Secondaries are 24v center tapped at 8 amps, 24v at 10 amps, and 24v at 10 amps! Possible combinations include 24v at 20 A and 48v center tapped at 10 amps. Shipping weight 13 lbs.
STOCK NO. B9898 $14.95 each, 2/28.00

DATA INPUT KEYBOARD HOUSING



This high impact plastic housing was part of a famous name programmable desktop calculator. It is tan, has the general appearance of a typewriter case, and is ideal for many types of computer keyboards. Mounting center for the keyboard is 15¼". The readout space is 15" x 4". Overall size is 18" x 18½" x 6½". 3 lbs. Will easily house an entire TVT or micrpcomputer. Prepunched hole in rear for a muffin fan. The opening in the top may vary from the one shown in the drawing.
STOCK NO. B9176 $3.75 each, 2/7.00

COMPUTER GRADE ELECTROLYTICS

We carry a good selection of capacitors of all types, dipped micas, ceramic disks, mica trimmers, tantalums, silvered micas, tubulars, small electrolytics, high voltage, etc. Listed below are a few of our large computer grade electrolytic capacitors.

VOLT	MFD	STOCK	PRICE	
10v	40,000	B2026	2.00 each,	6/11.00
10v	90,000	B2495	3.00 each,	3/8.00
10v	160,000	B2515	3.25 each,	4/12.00
15v	110,000	B2352	3.50 each,	3/9.00
25v	32,000	B2492	3.00 each,	4/10.00
35v	40,000	B2255	3.50 each,	3/9.00
50v	10,000	B2493	3.25 each,	4/12.00
75v	6,000	B2450	3.50 each,	4/12.00
200v	500	B2345	2.50 each,	4/9.00

AUTOTRANSFORMERS

120v to 240v step-up or 240v to 120v step-down autotransformer, 150 watts. 3 lbs. STOCK NO. B9844 $3.50 each, 3/9.00

2 KVA autotransformer. Taps at 105, 110, 115, 120, & 125 volts. Ideal for boosting voltage during brown-outs for TVs, etc.
STOCK NO. B9861 11 lbs. $8.95 each, 2/16.00

DELTA ELECTRONICS CO.

BOX 1, LYNN, MASSACHUSETTS 01903
Phone (617) 388-4705

MINIMUM ORDER $5.00. Include sufficient postage, excess refunded. Send for new 88 page Catalog 15, bigger than ever.

BANKAMERICARD and MASTERCHARGE now accepted, minimum charge $15.00. Please include all numbers Phone orders accepted.

BYTE'S BITS

Here's the Official Announcement: PL/M6800

Intermetrics, Inc, of Cambridge MA announces that PL/M6800,TM the first high-level programming language for the widely used Motorola 6800 microcomputer, is available.

The PL/M6800 compiler is now accessible for world-wide use on the General Electric Information Services computer network, or can be purchased directly from Intermetrics for in-house installation on IBM 360 or 370 computers.

According to an Intermetrics press release, "PL/M6800 is an effective aid to microprocessor software development. It brings the benefits of high-level language programming to the field of microprocessor programming. PL/M6800 cuts programming time and effort to a fraction of that required for assembly language, while allowing access to hardware features and assembly language procedures." This results in a reduction of programming cost and time. The language encourages the use of structured programming techniques, and is self-documenting.

PL/M6800 has a one-pass compiler which produces optimized object code in a format directly usable by the Motorola MINIBUG/MIKBUG and EXbug Loader Function. The compiler includes a number of user-controlled features, such as: source program listing, object code listing, equivalent assembly code listing, and symbol table dumps.

The PL/M6800 compiler is compatible with the PL/M language developed by Intel to program their line of microprocessors. Intermetrics reports that PL/M programs can be compiled with few, if any, changes, thus allowing a "significant degree of software portability."

Intermetrics has been engaged in computer software design and implementation for over six years. Its computer language developments include HAL, which is being used to program the NASA Space Shuttle on-board computers, as well as CS-4 and SPL/I, general purpose, high-level languages.

Full information and documentation on PL/M6800 is available from Marketing Department, Intermetrics, Inc, 701 Concord Av, Cambridge MA 02138.

MITS ALTAIR CONVENTION

MITS has announced a "World Altair Computer Convention," scheduled for the last weekend of March, to be held at the new MITS headquarters near the Albuquerque Airport, Albuquerque NM. According to information obtained from MITS, the convention will include a weekend of seminars and demonstrations conducted by MITS engineers and software writers. MITS system owners are also encouraged to bring systems for demonstration: Prizes worth several thousand dollars will be awarded for the best demonstrations by users in several system categories. For further information, contact:

WACC/MITS
2450 Alamo SE
Albuquerque NM 87106
Attn: David Bunnell

or call David Bunnell (505)255-2206.

Classified Ads Available for Individuals and Clubs

Readers who have equipment, software or other items to buy, sell or swap should send in a clearly typed or printed notice to that effect. The notices are free of charge and will be printed one time only on a space available basis. Insertions should be limited to no more than 100 words. Notices can be accepted from individuals or bona fide computer users' clubs only. Commercial advertisers should contact Virginia Peschke at BYTE for the latest rate card and terms.

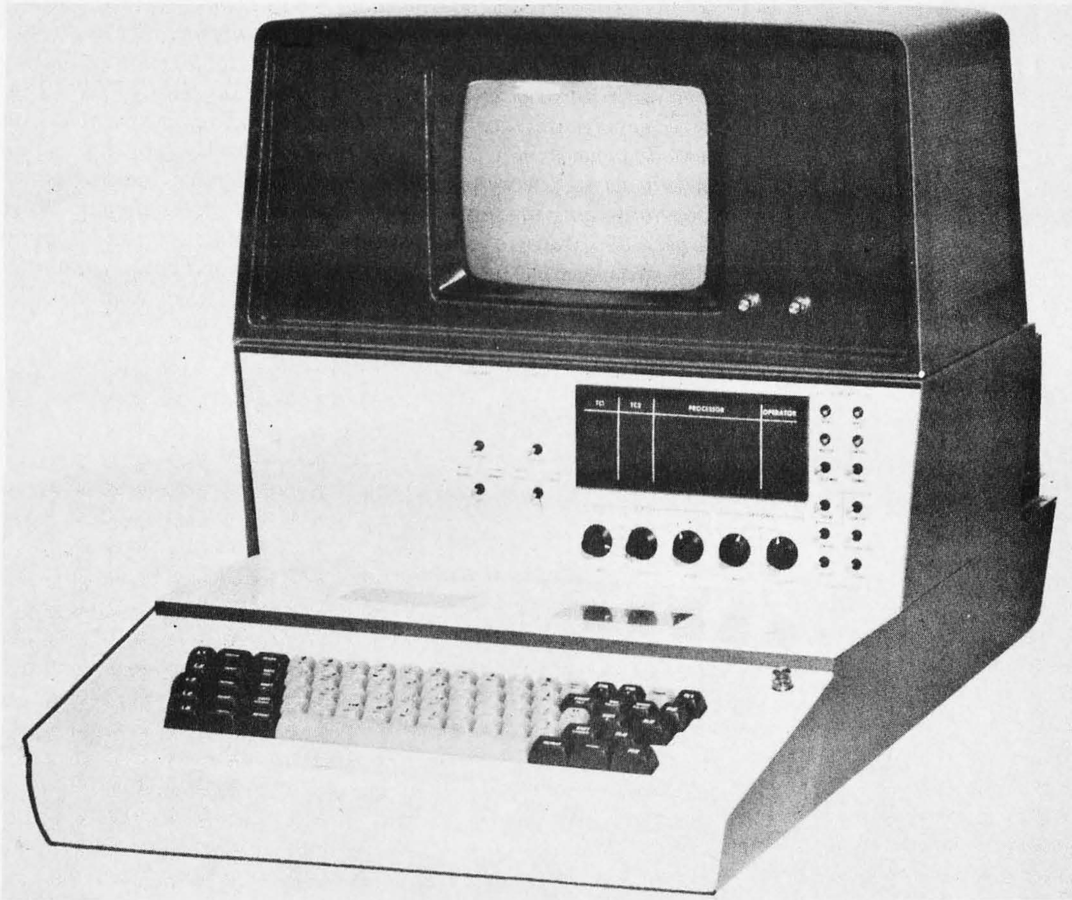
I am currently configuring a home computer system around a BIT 480 mainframe. If you own a similar machine and (a) have the documentation for the BIT 480 Diagnostics, (b) know of a users group, or (c) would just like to swap notes, please contact me. Ron Emerson, MD, Alden Park Manor #207C, Philadelphia PA 19144.

FOR SALE: 8080 system, G-10 plated holes, 22K RAM, sockets for 10K ROM (1702A), assembly language, with highly regulated supply 10A-5V, -18 (variable) and +/– 24V. $900 or best offer or trade for a 16 bit machine. Bruce Lill, Box 248, Hohokus NJ 07423, (201)444-0908.

TERMINAL wanted. If anyone in the Washington DC area has a TV Typewriter, KSR or ASR Teletype in good condition at a reasonable price, please contact: Steve Pierce, 5100 38th St NW, Washington DC 20016.

FOR SALE: AMPEX 3DM-3000 Core. 10 cards, each 1Kx9 with all drive circuitry. TTL compatible, new condition. $50 per card or make offer for all 10. Schematics included. William Brown III, 2449 N Park Blvd, Santa Ana CA 92706.

FOR SALE: Six 14-channel tape transports with design data. Direct recording to 2MHz at 60 or 120 inches per second. Designed for one inch wide tape. For more information, contact Francis T Tortorello, General Delivery, Nellis AFB, NV 89191.



SYSTEM 21 DATA MANAGEMENT STATION

VIATRON'S System 21 is a family of data processing devices designed for data management, including data entry, control, display, communication, storage and retrieval. With its modular structure, System 21 can be configured to perform a wide variety of data processing operations.

A typical System 21 configuration includes a Microprocessor, two Tape Channels, a Keyboard, two Data Channels, and a Video Display. Central to the System 21 structure is the Microprocessor which contains hard-wired microprograms that perform a fixed set of logical operations. The hard-wired microprograms in the Microprocessor accomplish the same functions as a general-purpose computer operating system or assembler. Because the microprograms are hard-wired, however, there is no need for extensive programming.

There are two modes of system operation–manual control and program control. In the manual mode of operation, the operator initiates all Microprocessor functions. Under program control the Microprocessor performs certain functions automatically through the use of a control program.

The Microprocessor has four input/output channels, two Tape Channels and two Data Channels. The Tape Channels are devoted to either VIATAPE Recorders or Computer Tape Recorders, one recorder per channel. The two Data Channels can communicate with optional input/output devices. They can be connected, for example, to a Model 6001 Card Reader/Punch Adapter for reading and punching cards, or to a Model 6002 Printing Robot for providing hard copy. The Data Channels can also be interfaced with a Model 6003, 6004, or 6005 Communication Adapter for providing a link with another System 21 Data Management Station, a computer, or virtually any other device capable of USASCII interface. The Keyboard has its own Channel dedicated to providing data input and control to the Microprocessor.

Unused, packed in 4 cartons. System consists of video display, power supply, microprocessor, two cassette tape decks mounted in microprocessor panel, keyboard, all as pictured. Sold "as is." Due to 4 years of storage, may require some adjusting/cleaning. With instruction book. Shipment within 24 hours if paid by MC, BA, or certified check. Sold FOB Lynn Mass.

$ 425.00

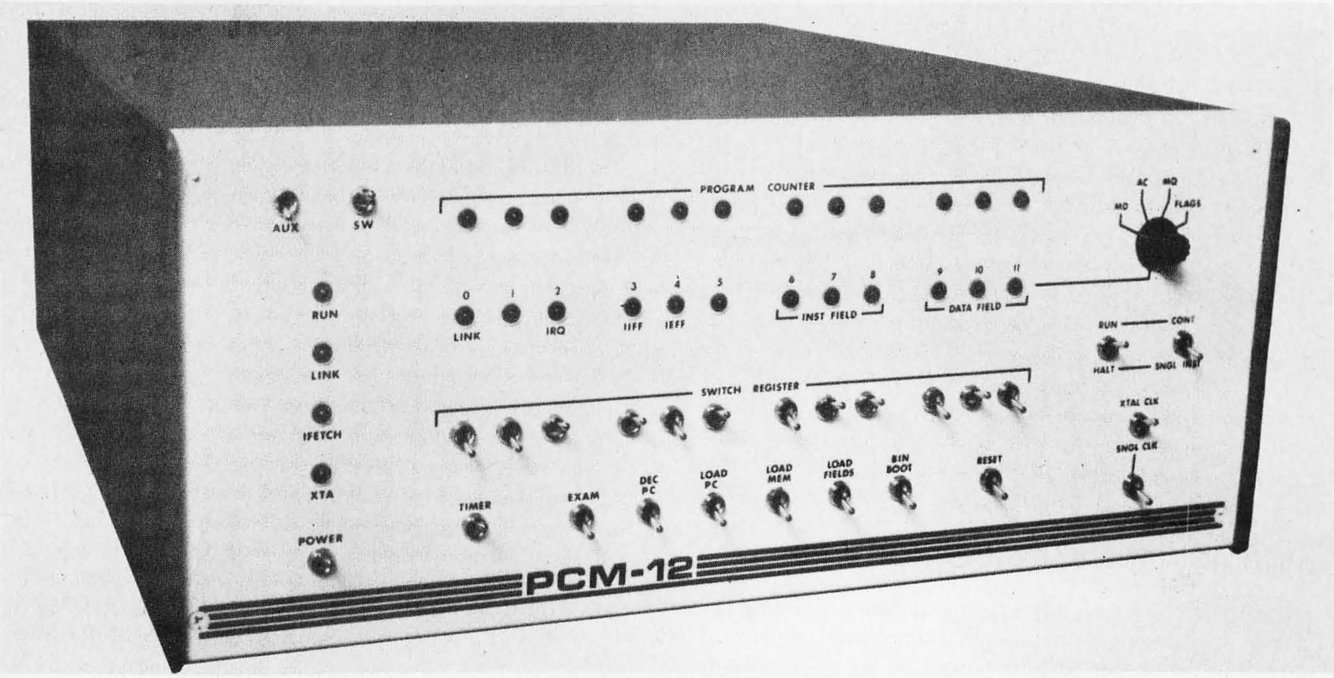
Meshna

MESHNA PO Bx 62 E. Lynn Mass. 01904

What's New?

THE FIRST OF THE 12 BIT MICROS?

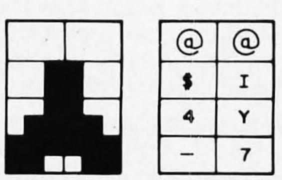
The PCM-12 is the first microcomputer product BYTE has seen built around the Intersil IM6100 computer chip. Quoting from the news release accompanying this photo, ". . . This kit is designed around the Intersil IM6100 microprocessor, a 12 bit static CMOS device that is software compatible with the Digital Equipment Corporation (DEC) PDP-8/E minicomputer. The completed kit can execute most PDP-8 software, including assemblers, editors, debug routines and advanced languages like BASIC and FORTRAN. Much of this software is available from DEC on an unlicensed, over-the-counter basis." Kit prices range from $400 to $600 depending upon options, and the result is the computer pictured here. If you order the complete kit, you get 4096 words of static semiconductor memory (12 bits per word), a serial terminal interface, audio cassette recorder interface, cabinet and power supply. The memory is expandable to 32,768 words. Contact PCM at PO Box 215, San Ramon CA 94583 or call (415)837-5400.



BYTE'S BUGS

Here lies documentation of known bugs detected in previous editions of BYTE . . .

The code for the pawn character in Don Lancaster's "Color Graphics" article, figure 8 (page 67, February), is incorrect as printed. The proper code is illustrated here:



DISPLAY ASCII

(C) PAWN

Note also that in the format definition of figure 4 (page 65), the most significant bit is on the right and the least significant bit is on the left.

Dr George Haller points out a misalignment of the internal captions in the UART diagram of figure 4, page 26, in Don Lancaster's "Serial Interface," September 1975. The correct captions for pins 34 to 38 should read:

Pin	Name
34	ENABLE
35	PARITY KILL
36	STOP BITS
37 } 38 }	BITS/CHARACTER

BYTE'S BITS

A 6800 CATERPILLAR PROGRAM

In a previous BYTE, we published an article on how to use blinking lights as action peripherals. ("There's More To Blinking Lights Than Meets The Eye," page 52, January 1976.) In that article, the CATERPILLAR program was described for an 8008 computer architecture. Here is a 32 bit version of CATERPILLAR which runs in a 6800 system, with the following assumptions:

1. Memory address space locations hexadecimal 00 to FF are implemented as random access memory.
2. Memory address space locations hexadecimal 014C to 014F are four display latches of the type described in "There's More To Blinking Lights Than Meets The Eye." 014C is the rightmost set of lamps, 014F the leftmost.
3. Memory address space locations greater than hexadecimal 1000 are implemented as random access programmable memory.

Start the 32-bit 6800 *CATERPILLAR by* branching to location 1000 after entering the program. Note that this routine is relocatable; this means that all branches are made using the relative addressing mode of the 6800, and the data is kept separate from the program in locations found in the first page of memory. Another subtle point is that the carry flag must be preserved within the main loop in order to maintain continuity. As written, the bug steadily moves from left to right in the display. As an exercise, try re-writing the program to cause the CATERPILLAR to move in the opposite direction. cth

```
1000  86  3F      CATERPLR  LDAA  #$3F      initial pattern of the worm;
1002  97  10                STAA  R0        save the worm in R0;
1004  7F  00  11            CLR   R1        clear rest of worm track
1007  7F  00  12            CLR   R2            found in the
100A  7F  00  13            CLR   R3            working variables of program;
100D  0D                    SEC             set carry (always have minimal worm);
100E  CE  03  FF  NEWMOVE   LDX   #$03FF    set up time delay count;
1011  09          WAITLOOP  DEX             delay loop sets speed of bug;
1012  26  FD                BNE   WAITLOOP  if not ready then continue delay;
1014  76  00  13            ROR   R3        this is a
1017  76  00  12            ROR   R2            32 bit
101A  76  00  11            ROR   R1            rotation through
101D  76  00  10            ROR   R0            the carry in work registers;
1020  96  10                LDAA  R0        fetch the right light pattern
1022  B7  01  4C            STAA  LAMP0         and store it in rightmost output;
1025  96  11                LDAA  R1        fetch right middle light pattern
1027  B7  01  4D            STAA  LAMP1         and store it in right middle output;
102A  96  12                LDAA  R2        fetch left middle light pattern
102C  B7  01  4E            STAA  LAMP2         and store it in left middle output;
102F  96  13                LDAA  R3        fetch leftmost light pattern
1031  B7  01  4F            STAA  LAMP3         and store it in leftmost output;
1034  20  D8                BRA   NEWMOVE   continue indefinitely;
```

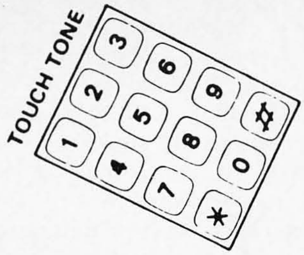
Symbol Table:

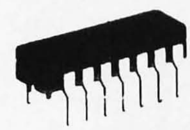
Data Locations

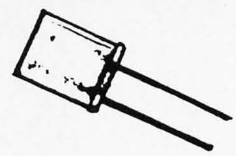
LAMP0	014C	Rightmost	display lamps
LAMP1	014D	Right middle	
LAMP2	014E	Left middle	
LAMP3	014F	Leftmost	
R0	0010		working RAM registers
R1	0011		
R2	0012		
R3	0013		

Program Locations

CATERPLR	1000	entrypoint of program
NEWMOVE	100E	main loop return point
WAITLOOP	1011	time delay wait return point







TOUCH TONE ENSEMBLE

12 Key pad (2.25" x 3") **plus** MC14410P Generator chips **plus** 1 MHZ crystal. Includes spec for MC14410 with circuit diagrams.

TTE-0121	$18.45

VOLTAGE REGULATORS

JUST OUT! LM317K 1.5 Amp 3 terminal adjustable regulator in TO-3 case. Adjusts from +1.2 to +37 V. Current limit and thermal shutdown.

LM317K	$ 4.99
Specs and applications	$ .80

78HO5 – 5V, 5A fixed voltage regulator in TO-3 case. Just plug it into same socket as your 309K and get 5 amps capability.

78HO5	$10.95
Specs	$ .30

T0-92 regulators. Tiny plastic fixed voltage regulators for up to 100 mA. Save space and money. 78 series positive. 79 series neg. last 2 digits indicate voltage.

78L05, 78L12, 78L15	$ .85
79L05, 79L12, 79L15	$ 1.00

CA3085A. 1.8V to 26 V adjustable. 100 mA. Short circuit protected.

CA3085A special	$.60 ea., 10/$5

CLOCKS AND TIMERS

MICRO POWER CLOCK

MC14440Z is a calendar and time of day clock chip designed for portable clocks and watches. Drives LCD display. Runs for months on single 1.5V cell.

MC14440Z	$14.75
Spec	$ .60

MLC400 Liquid Crystal Display for use with MC14440. Has 4 inch high characters.

MLC400	$10.00
Specs	$ .40

1CM7045. Precision timer chip for digital stop watch/clock/timer applications. On chip oscillator circuit. Stop watch can be standard, sequential, split or rally mode. 8 digit direct LED drive output indicates 100ths of seconds to 24 hours. Needs only LED array, crystal, switches and battery for operation.

1CM7045	$29.95
Specs	$ 1.00

1CM7207 and 7208 set for frequency counter. 7208 is 7 decade counter/decoder/driver. 7207 is oscillator/divider/controller.

1CM 7207-7208 set	$29.95
Specs	$ 1.70

COMPONENTS

1N4148 (on real tape)	20/$1.00
6V 400 mW zener diode	20/$1.00
1N4001	15/$1.00
1N4002	14/$1.00
1000V 2.5 Amp (IRC)	10/$2.00
100V 3 Amp EPOXY DIODE	10/$1.00
50V 3 Amp EPOXY BRIDGE	3/$2.00

COMPUTER COMPONENTS

1CM6100 12 bit micro processor chip recognizes PDP 8/E instruction set. C MOS construction-runs on micro amps.

1 CM 6100	$52.50
Data packet	$ 4.00

AY5-1013A High speed UART for use up to 40K BAUD.

UART-1013A w/specs	$10.95, 2/$19.95

MC14412 UNIVERSAL DATA MODEM CHIP.

Originate/answer, half/full duplex, sine wave out. Up to 600 BAUD.

MC14412 FL (4.75-15 Volts)	$28.99
MC14412 VL (4.75-6 Volts)	$21.74
Data	$ .90

MC14411 Bit rate generator. Crystal oscillator circuit and dividers generate Hertz to 1,843 MHZ data rates.

MC14411P	$11.98
Data	$ .60

1K static memory. 2102-1 (500 ns)

New, prime parts	$4.15, 10/$39.00

COMPUTER GRADE CAPS

5,400 µF	5 V	$ .60
720	150 V	$1.25
36,000	15 V	$2.25
19,000	25 V	$1.25
9,400	40 V	$1.10
56,000	10 V	$1.60
1,000	50 V (plastic)	$ .90
4,000	50 V (plastic)	$1.35
1,000	50 V (Twist Lok)	$ .65
100	250 V (Twist Lok)	$ .65

AXIAL LEAD ELECTROLYTICS

500 µF	25 V (Sprague)	10/$2.00
500	50 V (Temple)	$ .60
580	75 V (Sawgamo)	$ .65
1,000	25 V (ABC)	$ .60
4,000	25 V (CDE)	$ .95

SWITCHES

SPDT 1/4 inch toggle (removed)	3/$1.00
SPDT 1/4 inch sideways PC mount (new)	5/$4.00
SPDT super small slide switch	10/$2.00
DPDT Lighted Rocker SW	$1.49
SPDT 1/4 inch momentary-wire wrap	$ .99

MISC

Ferrite Beads 1/8" x 1/8" dia.	20/$1.00
IK CTS Open Micro Pot	10/$1.00
MK-20 TO-3 Kit (Anodized Al)	4/$1.00
12 V 100 mA P.C. Transformer	$1.29
S.P.N.O. 16A 12 V DC Coil Relay	$1.00

tri-tek, inc

6522 N 43rd Avenue
Glendale AZ 85301

All orders postpaid – please add insurance. Minimum order $5 U.S./$15 foreign. Latest lists – 13 cent stamp.



BOOK REVIEWS

The Missing Man *by Katherine MacLean, published by Berkeley Publishing, 1975, 220 pp, $6.95 hardback.*

Based on the Nebula award winning short story of the same name, Katherine MacLean tells of the future of The City, a large metropolitan area where people are slightly telepathic and are influenced by the trauma of a single individual. In order for panic to be kept under control, a sophisticated rescue squad is dispatched to reach an endangered person before his thoughts can induce mass hysteria.

George Sanford, the main character, finds himself particularly adapted to this sort of thing and is instrumental in locating a missing computer technician of the city's maintenance prediction department. The technician falls in with a group of revolutionaries who put his knowledge of the city's computers to use simulating equipment failures, a sophisticated form of sabotage.

In searching for the missing computer technician, George meets the genius leader of the gang and is pulled into the gang's actions; George himself becomes the missing man.

The book is good reading for those interested in speculating on the effects of computer control of urban areas.■

Digital Design With Standard MSI & LSI *by Thomas R. Blakslee, John Wiley & Sons, 1975, 357 pp, $19.95 hardback.*

There are now quite a few texts on digital design available, but this is the first I have seen that concentrates on minimizing IC packages rather than gate inputs and flip flops. There are many useful areas covered, including tradeoffs between digital and other electric or mechanical components, basic design theory, and reliability.

The first two chapters cover design considerations as a whole and establish the argument that traditional gate minimization is not only time consuming but usually more costly. Chapters 3 to 5 cover traditional design combined with the idea of using MSI and LSI wherever possible. Chapters 6 and 10 are very helpful essays on design problems that are usually ignored by traditional texts. Race conditions, noise, reflections, and clocking problems are treated in a clear manner.

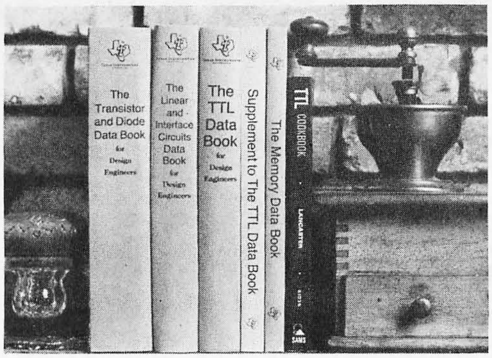
Chapters 7 and 8 look at programmed logic and cover the Intel 8008 and IMP-16 processors and instruction formats. The presentation is done in a way that easily applies to other microprocessors. Concepts of DMA, interrupts, assemblers and compilers, and automatic testing are well presented.

Chapter 9 discusses time multiplexing functions using a single circuit, and Chapter 11 deals with interfacing a wide variety of IO devices. Chapter 12 is a valuable reference for figuring system reliability using statistical calculations.

Perhaps most unusual about the text is chapter 13 which discusses the engineer's responsibility to society — will LSIs be used to create more and more consumer gadgetry or really help man to overcome his problems?

This is a well written and valuable book that belongs in every microprocessor library.■

Gary Liming
3152 Santiago Dr
Florissant MO 63033



You'll Want to Nybble at these Byte Books

Where does the editor of a computer magazine turn to when he must verify some author's hardware design? Information on a 75450 interface gate, or a 74147 priority encoder circuit does not spring forth by magic. Checking the information supplied by authors is part of BYTE's quality control program.

When you build a project, you need this same sort of information. All you find in the advertisements for parts are mysterious numbers identifying the little beasties . . . hardly the sort of information which can be used to design a custom logic circuit. You can find out about many of the numbers by using the information found in these books. No laboratory bench is complete without an accompanying library shelf filled with references – and this set of Texas Instruments engineering manuals plus Don Lancaster's *TTL Cookbook* will provide an excellent starting point or addition to your personal library.

- **The TTL Cookbook** by Don Lancaster, published by Howard W. Sams, Indianapolis, Indiana. Start your quest for data here with Don's tutorial explanations of what makes a TTL logic design tick. 335 pages, $8.95 postpaid.

- **The Supplement to The TTL Data Book for Design Engineers,** by Texas Instruments Incorporated. What happens when you can't find a 7400 series device listed in **The Data Book for Design Engineers?** Before you start screaming and tearing your hair out in frustration, turn to the Supplement. The Supplement has 400 pages of additional information including a comprehensive index to both TTL Data Book volumes. To be complete (and keep your hair in place and vocal cords intact) you'd best order the supplement at $1.95 to accompany the main volume.

- **The Transistor and Diode Data Book for Design Engineers,** by Texas Instruments, Incorporated. You'd expect a big fat data book and a wide line of diodes and transistors from a company which has been around from the start of semiconductors. Well, it's available in the form of this 1248 page manual from T.I. which describes the characteristics of over 800 types of transistors and over 500 types of silicon diodes. This book covers the T.I. line of low power semiconductors (1 Watt or less). You won't find every type of transistor or diode in existence here, but you'll find most of the numbers used in switching and amplifying circuits. Order your copy today, only $4.95 postpaid.

- **The Semiconductor Memory Data Book for Design Engineers,** by Texas Instruments, Incorporated. Don't forget the importance of memories to your systems. Refer to this 272 page manual to find out about the T.I. versions of many of the popular random access memories and read only memories. Order your personal copy today, only $2.95 postpaid.

- **The Linear and Interface Circuits Data Book for Design Engineers,** by Texas Instruments Incorporated. When you run across one of those weird numbers like 75365 the immediate frustration problem occurs again. What kind of gate could that be? We won't tell in this ad, but you can find out by reading the specifications in **The Linear and Interface Circuits Data Book for Design Engineers.** You can interface your brain to the 72xxx (linear) and 75xxx (interface) series of functions by ordering your copy of this 688 page manual at only $3.95 postpaid.

- **The TTL Data Book for Design Engineers,** by Texas Instruments Incorporated. How does an engineer find out about the TTL circuits? He reads the manufacturer's literature. This 640 page beauty covers the detailed specs of most of the 7400 series TTL logic devices. No experimenter working with TTL has a complete library without **The TTL Data Book for Design Engineers.** Order yours today, only $3.95 postpaid.

Caution: Heavy Reading

Buyers of these books should be cautioned: heavy reading will be required. These books are so filled with information that they weigh in at a total of about 190 ounces (5387 grams). On the basis of sheer mass, these books have got to be the bargain of the century. Make sure that you use a structurally sound book shelf and above all avoid dropping one of these books on your foot. But the mass of these books doesn't affect the bargain: we pay postage on all orders shipped to addresses in the USA and Canada, so the prices you see are the prices you pay. (That's only $.005 per gram on the average.)

____ TTL Cookbook @ $8.95
____ TTL Data Book @ $3.95
____ Supplement to TTL Data Book @ $1.95
____ Linear and Interface Circuits @ $3.95
____ Semiconductor Memory Data @ $2.95
____ Transistor and Diode Data Book @ $4.95

Send to: Name ________
Address ________
City ____ State ____ Zip ____
☐ Check enclosed
☐ Bill MC # ____ Exp. Date ____
☐ Bill BA # ____ Exp. Date ____
Signature ________

BYTE PETERBOROUGH, NH 03458

A Lesson in Economics

What happens when a good becomes so desirable and sought after that many people want to have it? Unless an ample supply is forthcoming, the price can and must go up (political fiat to the contrary notwithstanding). Recently we wondered why one distributor kept asking us for more and more of BYTE Number 1. He was very insistent, and spent several transcontinental telephone calls urging us (in vain) to send him more of BYTE's inaugural issue.

We found out recently, thanks to several callers, that this fellow was selling BYTE's Number 1 issue for $15.00 each, since everybody wants one and the supply is very limited. Of course he made a temporary windfall profit on his early investment in a supply of the new magazine. He paid what everyone else did who purchased a bulk shipment of the then unknown magazine. There is a law of economics that price rises when supplies are tight – it is as much a folly to fight such a trend as it is to attempt to legislate Pi = 3.0000. . . Many thanks to Adam Smith and Ludwig von Mises for tipping us off about the law of supply and demand, confirmed again by this laboratory experiment.

The point of mentioning this is to put to rest rumors about our supposed reprinting of the early issues of BYTE. First, we are not planning to reprint BYTE magazine issues as magazines, ever. You'll see individual articles reappearing in books of selected reprints which we plan to publish. However our early supporters will see their confidence rewarded as the price of the first issue collector's item rises over the years, since we will never reprint it as a magazine. Our policy is now and will continue to be one of matching our press runs to the subscription and newstand demand, with very few extra issues kept available. If you want to keep up with this fast moving field, you'll have to keep your subscription current.

Cut

Name ....

Address ....

City .... State............ Zip.......

☐ BILL ME ☐ Check for $12 enclosed
☐ Bill BankAmericard or MasterCharge # ....

Tear

Name ....

Address ....

City .... State............ Zip.......

☐ BILL ME ☐ Check for $12 enclosed
☐ Bill BankAmericard or MasterCharge # ....

Shred

Name ....

Address ....

City .... State............ Zip.......

☐ BILL ME ☐ Check for $12 enclosed
☐ Bill BankAmericard or MasterCharge # ....

Rip

Name ....

Address ....

City .... State............ Zip.......

☐ BILL ME ☐ Check for $12 enclosed
☐ Bill BankAmericard or MasterCharge # ....

• BYTE • 70 Main St • Peterborough NH 03458

Continued from page 8

files which contain images of a program written in a high level language; the memory image form of a program is an example of general binary data encoded in a way which has significance as a machine language representation of the program. When you design a particular application, choice of the data formats and representations to be used with mass storage is often one of the biggest considerations.

The Generality of IO Software

The fact that programs stored on a mass storage medium are simply data patterns with particular significance leads to the suggestion that the same IO software can be used to locate and transfer data for any purpose. The IO software can be looked upon as a channel through which bytes can be pushed (or from which they can be pulled). The specific meaning of the bytes transferred is for the most part independent of the use of the data. To illustrate this generality, take the specific example of the read and write software of the COMPLEAT Tape Cassette Interface in this issue. All this software does is transfer eight bits of data to the accumulator from the ACIA, or vice versa. The value of the byte does not matter. The generality shown at the byte level in that software can also be designed into the software for more sophisticated multiple byte records. The techniques of naming files mentioned earlier is such an extension which can be made with complete generality; other information management concepts also can be used. Indeed this barely scratches the surface of the concepts of files and data organization as implemented in IO software. The subject is worthy of several articles (wouldbe authors take notice).

The Status of Inexpensive Mass Storage Technology

As of the date of this writing (early January 1976) there exists only one truly inexpensive mass storage medium: The audio signals recorded and played back by typical cassette recorders or *any other audio signal medium*. (Note that the BYTE Audio Cassette Symposium's provisional standard uses the audio cassette recorder as a worst case design criterion; use of a better audio medium can only improve reliability.) At the present time, the primary use of the standard is for magnetic recording upon cassette or reel-to-reel recorders. There is presently no such thing as an inexpensive random

BYTE reader service

To get further information on the products advertised in this issue of BYTE merely tear, rip, or snip out this advertiser index, fill out the data at the bottom of the page, circle the appropriate numbers, and send the works to BYTE, 70 Main St, Peterborough NH 03458. Readers get extra Brownie Points for sending for information since this encourages advertisers to keep using BYTE – which in turn brings you a bigger BYTE.

ADVERTISER INDEX

Reader's Service
BYTE
Green Publishing Inc
70 Main St
Peterborough NH 03458

Please print or type.

MARCH 1976

BYTE acquired via
☐ Subscription
☐ Newsstand
☐ Stolen

Name ______________________________

Address ______________________________

City ______________ State ______ Zip ______

Coupon expires in 60 days . . .

access magnetic recording method. (Floppy disks are not in the same price class as audio cassette technology . . . yet.) One can only hope that engineers such as those who designed Hewlett-Packard's miniature magnetic card drives will come up with an equally reliable and inexpensive miniature random access medium for use in personal computing.

While the truly inexpensive random access mass storage device has yet to be designed, the sequential access represented by the low speed audio cassette storage medium is an existing and accomplished fact, as evidenced by the information in this BYTE. It is a quite usable system, even if slow; it will be present in amateur computing for some time to come.■

December's BOMB Winner

The results of the December BOMB analysis found the winner to be Don Lancaster's article on "Read Only Memory Technology." Runnersup were "Flip Flops Exposed" by William E. Browning and Gary Kay's article, "Build a 6800 System With This Kit." Thus Don receives the $50 bonus check as he did for his "Ins and Outs of Volatile Memory," in the November BYTE (*not* October, which last month's calendar confused BOMB results stated).

BOMB: BYTE's Ongoing Monitor Box

BYTE would like to know how readers evaluate the efforts of the authors whose blood, sweat, twisted typewriter keys, smoking ICs and esoteric software abstractions are reflected in these pages. BYTE will pay a $50 bonus to the author who receives the most points in this survey each month. (Editor Helmers is not eligible for the bonus.) The following rules apply:

1. *Articles you like most get 10 points, articles you like least get 0 (or negative) points – with intermediate values according to your personal scale of preferences.*
2. *Use the numbers 0 to 10 for your ratings, integers only.*
3. *Be honest. Can all the articles really be 0 or 10? Try to give a preference scale with different values for each author.*
4. *No ballot box stuffing: Only one entry per reader!*

Fill out your ratings, and return it as promptly as possible along with your reader service requests and survey answers. Do you like an author's approach to writing in BYTE? Let him know by giving him a crack at the bonus through your vote.

	LIKED LEAST										LIKED BEST
10 Hemenway: COMPLEAT Interface	0	1	2	3	4	5	6	7	8	9	10
18 Manly: Magnetic Recording	0	1	2	3	4	5	6	7	8	9	10
30 Lancaster: BIT BOFFER	0	1	2	3	4	5	6	7	8	9	10
40 Mauch: Digital Data	0	1	2	3	4	5	6	7	8	9	10
46 Baker: Update: CP1600	0	1	2	3	4	5	6	7	8	9	10
52 Helmers: Hand Assembly	0	1	2	3	4	5	6	7	8	9	10
62 Maurer: Algebraic Expressions	0	1	2	3	4	5	6	7	8	9	10
78 Walters: Video Display	0	1	2	3	4	5	6	7	8	9	10

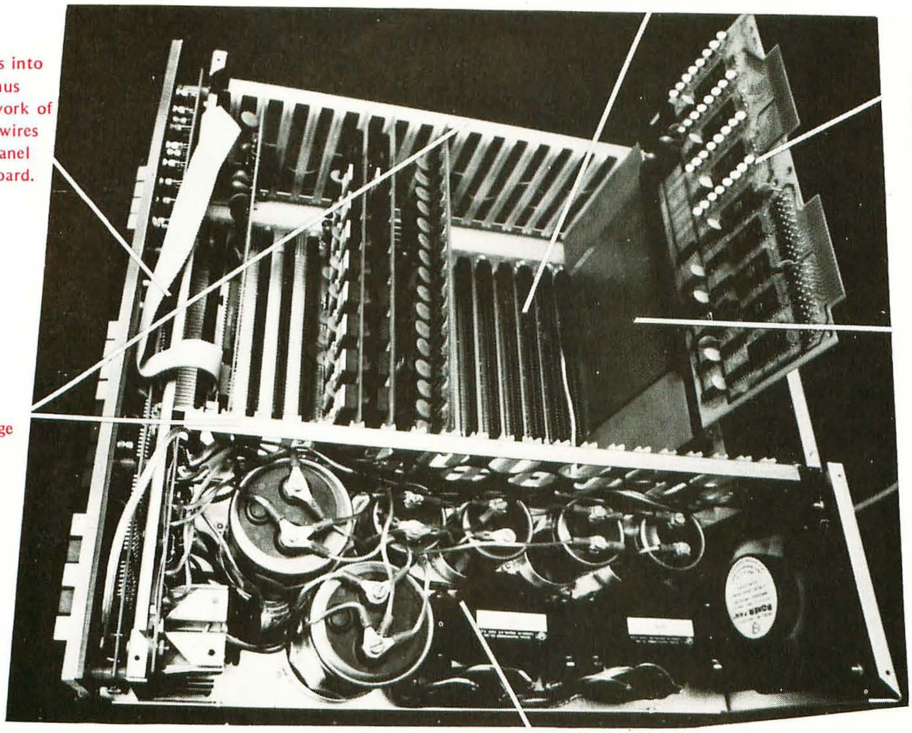
Feel free to photocopy this or any other page if you wish to keep your BYTE intact.

IMSAI

The IMSAI 8080 can be configured with optional Mother Board to provide a full 22 slots. (shown)

Front panel plugs into Mother Board, thus eliminating the work of soldering all the wires from the front panel to the Mother Board.

Special PIO Board configured with LEDs to monitor each bit of each of the 4 ports.



Extender Board available

Sturdy Card Cage

Heavy duty power supply (optional dual power supply shown.)

The IMSAI 8080 is designed using the full Intel family of large scale integration chips, thus providing high reliability and greater flexability.

Front panel hosts a photographic legend to produce a clear, concise, easy-to-read format that can be configured for either hexadecimal or octal. (It won't wear off!)

8 additional program controlled LEDs.



Commercial grade cabinet

Easy-to-use commercial grade paddle switches-- very crisp and solid.

Front panel unplugs, so unit can be used in a turn key system.

COMING SOON: Free BASIC and extended BASIC for registered IMSAI 8080 owners, followed shortly by Fortran IV and PLM.

TERMS: Check or money order, Bankamericard, Master Charge, 25% deposit on COD orders. On all orders under $1,000, add 5% for handling. On orders over $10,000 subtract 5%. California residents add 6% sales tax.

SEND FOR FREE CATALOG OF IMSAI MICROCOMPUTER PRODUCTS DEALER INQUIRIES INVITED

SPECIAL NOTICE TO ALTAIR 8800 OWNERS:

If you would like to step-up to the superior quality of an IMSAI 8080, you will be pleased to know that your ALTAIR 8800 boards are "plug-in" usable—without modification—in the IMSAI 8080 cabinet. Furthermore, by acquiring IMSAI's unique Memory Sharing Facility, your ALTAIR MPU board and IMSAI MPU board can co-exist in the same cabinet, operate in parallel with each other, and share all memory in common. This is the technology that laid the foundation for IMSAI's powerful HYPERCUBE Computer and Intelligent Disk systems (recently featured in Computerworld, Datamation and Electronics magazines.)

IMS ASSOCIATES, INC., 1922 REPUBLIC AVENUE, SAN LEANDRO, CA 94577

(415) 483-2093

If Napoleon had owned an Altair.



Things might have turned out differently.